Workgroup: PANRG
Internet-Draft:
draft-garciapardo-panrg-drkey-02
Published: 12 January 2022
Intended Status: Informational
Expires: 16 July 2022
Authors: J. Garcia-Pardo
ETH Zuerich
C. Kraehenbuehl
ETH Zuerich
B. Rothenberger
ETH Zuerich
A. Perrig
ETH Zuerich

Dynamically Recreatable Keys

Abstract

DRKey is a pragmatic Internet-scale key-establishment system that allows any host to locally obtain a symmetric key to enable a remote service to perform source-address authentication, and enables first-packet authentication. The remote service can itself locally derive the same key with efficient cryptographic operations.

DRKey was developed with path aware networks in mind, but it is also applicable to today's Internet. It can be incrementally deployed and it offers incentives to the parties using it independently of its dissemination in the network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 July 2022.

Table of Contents

1. Introduction

In today's Internet, denial-of-service (DoS) attacks often use reflection and amplification techniques enabled by connectionless protocols like DNS or NTP and the possibility of source-address spoofing. The main goal of DRKey is to provide a highly efficient global first-packet authentication system. DRKey provides packet authentication at the network layer based on the network address (i.e., the IP address in the current Internet or the combination of AS number and local address in SCION), and not based on a higher-level identity such as a domain name or web-server identity.

To obtain strong guarantees with high efficiency on a per-packet basis, an authentication system based on symmetric cryptography is required. DRKey does not rely on in-band protocols to negotiate keys, so it is able to authenticate already the first packet received from a host. DRKey also does not store the symmetric keys for all potential senders, as it would be infeasible in an Internet-scale system.

The core property achieved by DRKey is to enable a service to rapidly derive a symmetric key to perform network-address authentication for an arbitrary source host. This enables services such as DNS or NTP to instantly authenticate the first request originating from a client, thus providing a defense against reflection-based DoS attacks. The key can also be used to authenticate the payload of the request and reply, which is particularly useful for DNS which by default does not include any authentication.

The prototype system enables the server to derive the symmetric key within two AES operations, which corresponds to 18 ns on a commodity server platform, and authenticate the first packet within 85 ns on commodity hardware. Such speeds cannot be achieved with protocols based on asymmetric cryptography that require multiple messages to be exchanged to establish a shared session key. For example, DRKey outperforms RSA 1024-based source authentication by a factor of more than 220, even under the assumption that the service already knows the client's public key. In addition to providing highly efficient network address verification, DRKey can also be used to authenticate Diffie-Hellman (DH) keys in a protocol such as TCPcrypt.

1.1. Outline

The main ideas behind DRKey are as follows. Autonomous systems (ASes) can obtain certificates for their AS number and IP address range from a public-key infrastructure (PKI), i.e., SCION's control-plane PKI in a SCION deployment or the Resource Public Key Infrastructure (RPKI) in today's Internet. DRKey uses such a PKI to bootstrap its own symmetric-key infrastructure. DRKey key servers are set up in all deploying ASes and contact each other on a regular basis to set up symmetric keys between pairs of ASes. These symmetric keys are then used as a root keys to efficiently derive a hierarchy of symmetric per-host and per-service keys. The hardware implementation of the AES block cipher on most modern CPUs (Intel, AMD, ARM), allows such a key derivation in about four to seven times faster than a single DDR4 DRAM memory fetch. The approach described ensures rapid key derivation on the server side, whereas a slower key fetch is required by the client to a local key server. This one-sidedness makes the source authentication for the receiving side as efficient as possible and ensures that DRKey does not introduce new

DoS attack vectors. DRKey is incrementally deployable and provides immediate benefits to deploying entities.

A fundamental tradeoff in DRKey is the additional trust requirements of end hosts in their local AS: as the key server is able to derive the end-to-end symmetric key, this key cannot be used directly to achieve secrecy between two end hosts. However, DRKey keys can be used to authenticate that the source host indeed belongs to the claimed AS, which suffices to resolve DoS attacks.

2. Terminology

AS: Autonomous System. A one-entity managed network.

SCION: A Path-Aware inter-networking architecture.

Network Node: An entity that processes packets.

Key Server: An entity connected to the network, that contains cryptographic keys, and is able to provide such keys to their respective hosts, granted they have the required permissions.

End Host: A node in the network that executes programs in behalf of users. Users usually have full control of their end hosts.

PRF: Pseudo Random Function. Function that has a low time complexity to evaluate, but which inverse is very expensive to obtain, making it infeasible to compute. PRF may have as parameters a key and a value to which the function is applied.

DRKey Secret Value: A sequence of bytes kept in secret by the AS, inside the Key Server. The validity of the secret value is configurable per AS, and dictates the validity of other keys derived from it. The secret value is either random, or derived via a PRF from a random or secret sequence of bytes only known by the AS. Secret values are the root of the DRKey key hierarchy. A secret value for AS A is denoted as SVa. More generally, a secret value can be bound to a standard protocol p (denoted as SVa^p). Non-standard protocols do not have their own secret value.

DRKey Key Arrow Notation: In DRKey, level 1 and level 2 keys exist to allow the authentication of the communication between one source entity a and one destination entity b. The key is derived by one side and copied to the other. The side that derives the key is the source of the arrow in the DRKey key notation. So the key K_{b->a} denotes a key that is derived at b's side and obtained on a's side, independently of the flow of the packets. The source side of the arrow is also called the "fast side", and the destination, the "slow side". The fast side is typically a server, and the slow side an end host.

DRKey Level 1 Key:
A key derived from a protocol bound secret value, by specifying the source and destination AS IDs of the ASes involved in the communication. The level 1 key can be derived by applying a PRF keyed on the secret value, to the identifiers of the source and destination ASes of the derivation. A level 1 key between fast side AS A and slow one AS B is denoted as K_{A->B}^p for a standard protocol "p", of K_{A->B}^* for non-standard ones.

DRKey Level 2 Key: A key derived from a level 1 key, and used to authenticate the source of packets from end-hosts to infrastructure nodes, or to further derive level 3 keys. A level 2 key is derived by applying a PRF keyed on the level 1 key to the identifiers of the source and destination of the communication. These identifiers can be the AS ID plus the IP address for the slow side, and the AS ID or the AS ID plus the IP address for the fast side of the DRKey protected communication. All level 2 keys are anchored to a protocol, identified by a string. We distinguish two possible level 2 keys, depending on the fast and slow sides of the key. (1) A level 2 key between the fast side AS A and the slow side end host Hb in AS B for standard protocol "p" is denoted as K_{A->B:Hb}^p. (2) A level 2 key between the fast side endhost Ha in AS A and the slow side AS B for standard protocol "p" is denoted as K_{A:Ha->B}^p. For non-standard protocols the notation is the same but replacing p with *,p.

DRKey Level 3 Key: A key derived from a level 2 host-to-AS key, used to authenticate the source of end-host to end-host packets. A level 2 key between the fast side endhost Ha in AS A and the slow side end host Hb in AS B for standard protocol "p" is denoted as K_{A:Ha->B:Hb}^p. For non-standard protocols the notation is the same but replacing p with *,p.

MAC: Message Authentication Code is a sequence of bytes that authenticates and protects the integrity of a message. Modifying the sender identity or the content of the message is detected by the MAC.

3. Key Derivation

To convey an intuition of the operation of the DRKey system, a high-level overview is provided first.

3.1. Overview

The basic use case of DRKey is when a host Ha in AS A desires to communicate with a server Hb in AS B, and Hb wants to authenticate the network address of Ha using a symmetric key. ASes A and B have

set up one DRKey key server each, KSa and KSb respectively. Each AS randomly selects a local secret value, SVa and SVb, which is only shared with trustworthy entities (in particular the key servers) in the same AS. The secret values are never shared outside the AS. The secret value will serve as the root of a symmetric-key hierarchy, where keys of a level are derived from keys of the preceding level. In DRKey, the keys are derived using a CMAC with AES, which is an efficient pseudorandom function (PRF). The key derivation used by KSb in the example is: K_{B->A} = PRF_{SVb}(A).

Thanks to the key-secrecy property of a secure PRF, K_{B->A} can be shared with another entity without disclosing SVb. The arrow notation indicates the secret value used to derive the key. Thus K_{B->*} would typically be used if AS B is on the performance critical side, where * denotes the set of remote ASes.

To continue with the example, KSa will prefetch keys K_{*->A} from key servers in other ASes, including K_{B->A} from KSb. In the example, the server Hb is trustworthy, and can thus obtain the secret value SVb to derive keys quickly. When Ha wants to authenticate to Hb, it contacts its local key server KSa and requests the key K_{B:Hb->A:Ha}, which KSa can locally derive from K_{B->A}. Ha can now directly use this symmetric key for authenticating to Hb.

The important property of DRKey is that Hb can rapidly derive H_{B:Hb->A:Ha} by using SVb and performing two PRF operations. The one-wayness of the key-derivation function allows a key server to delegate key derivation to specific entities. The key derivation process exhibits an asymmetry, meaning that the delegated entity Hb can directly derive a required key, whereas host Ha is required to fetch the corresponding key from its local key server. As opposed to other systems that rely on a dedicated server for key generation and distribution (such as Kerberos), this delegation mechanism allows entities to directly obtain a symmetric key without communication to the key server.

3.2. Assumptions

- There exists an AS-level PKI, that authenticates the public key of an asymmetric key pair for each participating AS E and certifies its network resources; e.g. the SCION control-plane PKI certifying AS numbers for a deployment in SCION and RPKI certifying AS numbers and IP address ranges for a deployment in today's Internet.

- To verify the expiration time of keys and messages, DRKey relies on time synchronization among ASes with a precision on the order

of several seconds. This can be achieved using a secure time-synchronization protocol such as Roughtime.

* There exists an authentication mechanism for end hosts within an AS. This is needed for access control.

3.3. Key Hierarchy

The DRKey key-establishment framework uses a key hierarchy consisting of four levels:

* 0th-Level (AS-internal). On the zeroth level of the hierarchy, each AS A randomly generates a local AS-specific secret value SVa. The secret value represents the per-AS basis of the key hierarchy and is renewed frequently (e.g., daily). In addition, an AS can generate protocol-specific secret values: SVa^p = PRF_{SVa}("p") for a standard protocol p, where "p" is its ASCII encoding. The purpose of these values is that they can be shared with specific services, such as DNS servers, that cannot be trusted with SVa but should still be able to efficiently derive additional keys. This construction introduces additional communication and storage overhead, so only widely used protocols such as DNS or NTP would have their own secret values. Non-standard arbitrary protocols will not have their own independent secret value, and thus it won't be shareable among services. For these protocols, their level 1 keys will be derived from a special secret value denoted as SVa^*, only used for the derivation purpose.

* 1st-Level (AS-to-AS). By using key derivation, an AS A can derive different symmetric keys using a PRF from the special local secret value SVa^* or a protocol-specific secret value SVa^p. These derived keys, which are shared between AS A and a second AS B, form the first level of the key hierarchy and are called first-level keys: K_{A->B}^x = PRF_{SVa^x}(B). The input to the PRF is the (globally unique) AS number of AS B. The value of x will be either p for standard protocols or * for arbitrary ones. The first-level keys are periodically exchanged between key servers of different ASes.

* 2nd-Level (AS-to-host, host-to-AS). Using the symmetric keys of the first level of the hierarchy, second-level keys are derived to provide symmetric keys for authentication (AS-to-host cases) or further derivation (host-to-AS cases) into the third level keys. Second-level keys can be established between:

  - An AS as fast side, and an end-host as slow, for a standard protocol p: K_{A->B:Hb}^p = PRF_{K_{A->B}^p}(0||Hb)

-An end-host as fast side, and an AS as slow, for a standard protocol p: K_{A:Ha->B}^p = PRF_{K_{A->B}^p}(1||Ha)

-An AS as fast side, and an end-host as slow, for a non-standard, arbitrary protocol p: K_{A->B:Hb}^{*,p} = PRF_{K_{A->B}^*}(0||Hb||"p")

-An end-host as fast side, and an AS as slow, for a non-standard, arbitrary protocol p: K_{A:Ha->B}^{*,p} = PRF_{K_{A->B}^*}(1||Ha||"p")

*3rd-Level (host-to-host). These keys are derived from the second level host-to-AS, DRKeys. Depending on the protocol type, we observe two cases:

-Standard protocol p: the PRF is keyed on the level 2 host-to-AS drkey: K_{A:Ha->B:Hb}^p = PRF_{K_{A:Ha->B}^p}(Hb)

-Non-standard, arbitrary protocol p: the PRF is keyed on the level 2 host-to-AS drkey: K_{A:Ha->B:Hb}^{*,p} = PRF_{K_{A:Ha->B}^{*,p}}(Hb)

4. Key Establishment

There are two types of key establishment: first level, and second or third level key establishment, depending on the level of the key in the hierarchy.

4.1. First Level Key Establishment

Key exchange is offloaded to the key servers deployed in each AS. The key servers are not only responsible for first-level key establishment, they also derive second-level keys and provide them to hosts within the same AS. To exchange a first-level key, the key servers of corresponding ASes perform the key exchange protocol. The key exchange is initialized by key server KSb by sending the request:

req = A || B || val_time || TS || [p]

Where TS denotes a timestamp of the current time and val_time specifies a point in time at which the requested key is valid. If an optional protocol p is supplied, the protocol-specific first-level key K'_{A->B}^p is requested, otherwise the general K_{A->B} is. The requested key may not be valid at the time of request, either because it already expired or because it will become valid in the future. For example, prefetching future keys allows for seamless transition to the new key. The request token is signed with B's private key to prove authenticity of the request.

Upon receiving the initial request, KSa checks the signature for authenticity and the timestamp for expiration. If the request has not yet expired, the key server KSa will reply with an encrypted and signed first-level key derived from the local secret value SVa or, if an optional protocol p was supplied in the request, SVa^p:

```
key = PRF_{SVa}(B)
    or
key = PRF_{SVa^p}(B)

repl = {A || key}_{PK_B} || exp_time || TS
```

The term {A || key}_{PK_B} indicates that the concatenation of A with the key is encrypted with asymmetric cryptography using B's public key. The reply token is signed with A's private key.

Once the requesting key server KSb has received the key, it shares it among other local key servers to ensure a consistent view. Each key server can now respond to queries by entities within the same AS requesting second-level keys. Alternatively, the proposed first-level key exchange protocol could also make use of TLS 1.3 with client certificates to secure the key exchange.

All first-level keys for other ASes are prefetched such that second-level keys can be derived without delay. However, on-demand key exchange between ASes is also possible. For example, in case a key server is missing a first-level key that is required for the derivation of a second-level key, the key server initiates a key exchange. ASes that contain a large number of end hosts benefit from prefetching most first-level keys, as they are likely to communicate with a large set of destination ASes. In today's Internet there exist around 68000 active ASes. Thus, setting up symmetric keys between all entities requires minor effort and state. To avoid explicit revocation, the shared keys are short-lived and new keys are established frequently (e.g., daily). Subsequent key exchanges to establish a new first-level key can use the current key as a first line of defense to avoid signature-flooding attacks.

4.2. Second or Third Level Key Establishment

End hosts request a second-level key from their local key server with the following request format:

format = {type, requestID, protocol, source, destination}

An end host Ha in AS A uses this format for issuing the following request to its local key server KSa:

```
format || val_time || TS
```

It is assumed that this request and the reply are sent over a secure channel. Similar to the first-level key exchange, val_time specifies a point in time at which the requested key is valid. The key server only replies with a key to requests with a valid timestamp and only if the querying host is authorized to use the key. An authorized host must either be an end point of the communication that is authenticated using the second-level key or authorized separately by the AS.

If the end host requested a third level key, it must now be derived. It is done so from the obtained second level key.

4.3. Key Server Discovery

When a key server wants to contact another key server in a remote AS, it needs to know the IP address of the remote server.

In the SCION architecture, the concept of service addresses can be used to anycast to a key server in a specific AS.

For the regular Internet, RPKI can be used, which connects Internet resource information to a trust anchor. As the deployment numbers of RPKI are growing, the RPKI certificate can be extended with the IP address of the key server (e.g., by encoding it into the common name field or creating a separate extension). Using this mechanism, each AS publishes a list of IP addresses (or an IP anycast address) that is publicly accessible and shared by all key servers. The routing infrastructure will direct the packets to the topologically nearest key server. This mapping from an AS identifier to an IP address is verifiable through RPKI to prevent unauthorized announcements of key servers. In case RPKI has not been fully deployed, key-server discovery could also work using a DNS entry that maps a domain to IP addresses of key servers.

4.4. Key Expiration

Shared symmetric keys are short-lived (i.e., up to one day lifetime) to avoid the additional complication of explicit key revocation. However, letting all keys expire at the same time would lead to peaks in key requests. Such peaks are avoided by spreading out key expiration, which in turn leads to spreading out the fetching requests. To this end, a deterministic mapping offset (A, B) -> [0, t) is introduced. This function uniformly maps the AS identifiers of the source in AS A and the destination in AS B to a range between 0 and the maximum lifetime t of SVa. This mapping is computed using a (non-cryptographic) hash function:

```
offset(A,B) = H(A || B) mod t
```

The offset is then used to determine the validity period of a key by determining the secret value SVa^j that is used to derive K_{A->B} at the current sequence j as follows:

[start(SVa^j) + offset(A, B) , start(SVa^j+1) + offset(A, B))

I.e., depending on the destination B, the secret value SVa can be different, even when chosen for the same point in time.

5. Packet Authentication

The DRKey keys enable source authentication of every packet. To send DRKey source authenticated packets from end host Ha located in AS A to endhost Hb located in AS B, end host Ha first obtains the second level key K_{B:Hb->A}^p from the key server located in its AS A, KSa. With it derives the third level key K_{B:Hb->A:Ha}^p, which is used to authenticate. For a packet pkt, the source Ha then calculates the authentication tag by computing the MAC keyed on the third level key K_{B:Hb->A:Ha}^p, over an immutable part of the packet pkt. This immutable part of pkt can include parts of the layer-3 and layer-4 headers, and optionally the layer-4 payload. It is important to only include immutable fields as the verification would otherwise fail at the destination.

The packet received at the destination is used to determine the source address Ha and source AS.

- *In SCION these are part of the regular header, thus no extra information is needed other than the tag itself.

- *In the current internet, 4 bytes containing the AS ID, plus the tag are added to the packet.

The destination Hb then derives or obtains the key K_{B:Hb->A:Ha}^p and uses it with the same MAC function to recalculate the authentication tag. The tag is then compared to the one present in the packet.

5.1. High-Speed DNS Authentication

A protocol specific secret value is used SVb^p, with p = "DNS". The level 1 key for a slow side A is derived directly in the DNS server:

K_{B->A}^p = PRF_{SVb^p}(A)

This first level key is exchanged with other AS via the level 1 key requests, as described in Section 4.1. For a DNS query from a end host Ha, located in AS A, to a DNS server located in AS B, the first level key is derived as described above, and then the second level key is derived:

$$K_{B->A:Ha}^p = PRF_{K_{B->A}^p}(0 || Ha)$$

How to compute the authentication tag and obtain the AS ID is described in Section 5.

5.2. EDNS(0) Authentication Option

DRKey can use EDNS(0) to avoid breaking the existing DNS resolvers and authoritative servers. With a DRKey custom extension that includes the total query/response length, the source AS number, a timestamp, and the per packet MAC. The per-packet MAC for DNS queries and responses is computed the DNS header and all fields contained in the extension. Using the DRKey EDNS(0) option, packet authentication for every DNS packet introduces 28 bytes of header overhead.

6. Deployment

DRKey allows incremental deployment, as key servers could be gradually deployed in each AS. Already in the incremental deployment phase, DRKey prevents the addresses of upgraded ASes from being spoofed at other upgraded destination ASes. Early adopters can immediately profit from DRKey's security properties. Authenticating a packet requires packet modification either at the end host, or at a network appliance such as a middlebox or border router. Adding the MAC at the end host does not increase the request size en-route.

When updating end hosts is not possible in the short-term, DRKey can be implemented on a middlebox that computes a per-packet MAC and modifies applicable bypassing packets.

Packet verification at the destination AS can be performed by a middlebox as well. If a destination does not understand DRKey traffic, it could fail to process this traffic. In this case, the sending host contacts its local key server and asks if the destination AS supports DRKey. The key server might have previously derived second-level keys for an end host in the corresponding AS or might forward the query to a key server in the destination AS. If an AS supports DRKey, then it may deploy a middlebox that performs the DRKey operations in case the end host does not support it. This will prevent sending authenticated traffic to a destination host that does not support DRKey. In the worst case, the end host could fall back to legacy traffic.

In case of operational failures (e.g., a single key server fails), the end entity will try to contact another key server in the same AS. If all key servers fail, the system could fall back to the current system with unauthenticated traffic.

6.1. Deployment Incentives

Since DRKey can be deployed on commodity hardware and integrates well with the current Internet infrastructure, the deployment obstacle for DRKey is low. DRKey traffic can be recognized outside of ASes that have deployed DRKey and can thus be prioritized by servers. This means that even when relatively few companies deploy DRKey to authenticate packets at their services (e.g., popular open DNS resolvers of Google or Cloudflare), there are strong incentives for ISPs to deploy DRKey and provide its services to their customers.

6.2. Key-Server Latency

The initial connection setup depends on the latency of the connection between the client and the key server. To lower latency, DRKey's key servers should be positioned in an AS at a similar location as local DNS resolvers. As even public resolvers have an average query latency of less than 20 ms traversing the Internet, it is expected that the latency of a local key derivation will be in the order of a few ms. In most cases locally fetching a key will result in a lower latency than a full round-trip between client and server. For ASes that are geographically dispersed, multiple key servers may be deployed (e.g., co-located with an access router or per point-of-presence).

6.3. Network Mobility

Network mobility allows entities to move from one AS to another while maintaining communication sessions. In DRKey, key derivations are based on the current location of the entity in the Internet. Therefore, as soon as an entity moves to another AS, it needs to contact the key server of the new AS and fetch new second-level keys. Because local key derivation is fast and the latency of obtaining a key is small, the overhead is minimal.

6.4. Lighning Filter System as a DRKey Deployment

The Lightning Filter (LF) mechanism is a novel system that is intended to complement traditional firewalls by enabling cryptographically authenticated traffic shaping, based on the autonomous system of the source host of the traffic. This reduces significantly the load on the traditional firewall during denial-of-service attacks, and even allows LF to be the only network defense mechanism for a specific sub-network, e.g. by securing a DMZ that exposes external services to untrusted networks.

The core principle of the LF system relies on DRKey, using the high speed source authentication that DRKey enables. This way, the system

can authenticate each packet, assuring that it came from the host it claimed to.

In case a breach is detected, the network administrators can immediately add the host and/or the origin AS to a blacklist, preventing packets originating there from reaching past the Lightning Filter.

7. Security Considerations

7.1. DRKey and Trust in ASes

The keys provided by DRKey do not provide full end-to-end authenticity or secrecy properties: The source and destination ASes are able to derive the keys and could thus perform an active attack. This attack is limited to these two ASes; active attacks by intermediate ASes are not possible. DRKey always enables AS-level source authentication and host-level source authentication under the additional assumption of an honest source AS.

7.2. Authentication within an AS

To achieve secrecy as well as end-host authentication for communication between end hosts and key servers, an AS needs an intra-domain end-host and/or user-authentication system. Different authentication mechanisms based on the operational environment are discussed:

* Authentication using TLS. In order to securely exchange second-level DRKey keys between end hosts and key server, the end host can establish a secure TLS channel to the key server. The identity of the communicating parties is authenticated using public-key cryptography for both the key server and the end host. Thus, the key server can uniquely identify the end host and verify its legitimacy to obtain a second-level key.

* Deployment in ISPs. If the corresponding AS is an ISP, we assume that they can identify their customers (e.g., terminal connection number or router that has been deployed by the ISP). In this case, only an attacker that is present at the customers local network can gain access to learn keys.

* Company / University. For ASes that are under the control of companies or universities, login credentials or other local authentication mechanisms can be used to identify the user. This gives companies the ability to run their own web servers and have full control over their key material.

* Mobile Devices. For mobile devices such as smart phones that are connected to the Internet through a mobile telecommunication

network, clients could be authenticated by the telecom provider, for example using the International Mobile Equipment Identity (IMEI).

7.3. Adversary Model

The adversary can deviate from the protocol and attempt to violate its security goals. The Dolev-Yao model is assumed, where the adversary can reside at arbitrary locations within the network. The adversary can passively eavesdrop on messages and also actively tamper with the communication by injecting, dropping, delaying, or altering packets. However, the adversary has no efficient way of breaking cryptographic primitives such as signatures, pseudorandom functions (PRFs), and message authentication codes (MACs). It is assumed that there exists a secure channel between end hosts and a key server within the same AS.

Assuming the mentioned capabilities, the goal of the adversary is to obtain cryptographic keys of other parties to forge authenticated messages. By compromising an entity, the adversary learns all cryptographic keys and settings stored. The adversary can also control how the entity behaves, including fabrication, replay, and modification of packets. Both end hosts and network nodes compromises are considered.

8. IANA Considerations

This document has no IANA actions.

Authors' Addresses

Juan A. Garcia-Pardo
ETH Zuerich

Email: juan.garcia@inf.ethz.ch

Cyrill Kraehenbuehl
ETH Zuerich

Email: cyrill.kraehenbuehl@inf.ethz.ch

Benjamin Rothenberger
ETH Zuerich

Email: benjamin.rothenberger@inf.ethz.ch

Adrian Perrig
ETH Zuerich

Email: adrian.perrig@inf.ethz.ch