Extensible Authenticated Routing

Andres Erbsen Sandra Schumann

December 5, 2012

Abstract

This document gives a high-level overview of a routing scheme designed to accommodate connections that require different compromises between privacy and speed in the same network. Finding and choosing routes is separated from routing packets. This allows scalability issues to be addressed separately without compromising security and makes the system more flexible in general.

Goals and assumptions

The scheme described here is designed to be used as the primary means of digital communication of a community and therefore aims for extreme reliability and robustness. There is an inherent need to compromise between efficiency and privacy and we do not wish to force either extreme upon the users. Security should be pluggable and enabled by default, but not compulsory. Low-latency anonymity features are supported by the design, but possibly not implemented or offered by the devices[1].

The question how to find efficient routes in huge networks (comparable to the Internet) containing adversarial devices is yet to be solved. In fact, a *universal* solution may very well be impossible. For this reason, we push the route choice exclusively to the device that starts a connection and provide a starting point on which to build route finding schemes.

Devices in a network should be able to automatically assess the trustworthiness of other devices, achieving this functionality is yet another issue. There rarely are universally and absolutely trusted parties, so relying on them is not an option. Instead, using a web of trust is recommended, but not required to participate in a network. A device that does not trust anybody and is trusted by nobody else should still be allowed to participate and doing so must not compromise its security.

[1] Also: “node”, “station”

To achieve these goals, the use of public-key cryptography seems inevitable. There are schemes (cjdns[@cjdns]) that use only public-key encryption, but, as we observed[@cjdnsestimate], the additional constraints limit scalability. Our scheme requires both (many-time) digital signatures and authenticated public-key encryption.

The only requirement for the underlying medium is bidirectional unicast communication. If a broadcast channel is available, such as in WiFi networks, it can be made use of.

The details

Devices

The approach to device authentication is similar to the one used in PGP. The identity of a device is tied to its main signing key. As an encryption key is also required, it must be signed with the main key for it to be considered relevant. The same procedure can be used to add alternative keys.

Links

As a link between two devices is only useful to other participants of the network if the endpoints agree to forward somebody else's packages over it. Such a promise can be represented as a description of the link (latency, bandwidth, uni- or bidirectionality) and its endpoints signed by both parties [2]. Adding issue time to to these descriptions is critical, otherwise devices could (accidentally or because of a replay attack) confuse old and obsolete links with current ones.

As a special case, the link description representing absence of a link (dead link) has to be signed by only one party to be considered valid. There is no way to get the other device's signature if the link that was removed used to be the only connection to it. Also, if one of the endpoints decides not to service a certain link, it can just drop the packets heading over it without the other devices consent.

Routes are described as sequences of links. Having two routes, for example from A through M and N to X and another route from B through N through L to Y, a valid route from M to L through N can be constructed by any device by concatenating links from M to N in the first route with links from N to L from the second. As the link descriptions were not modified, the new route's trustworthiness can be checked by verifying the signatures on the links and does not depend on who composed it.

[2] A similar concept is used in SPROUT [@sprout]. Our work is independent of theirs.

Connections

To allow custom route finding, route choice must happen at the device who initiates a connection. The chosen route may not be short, so annotating every packet with it would cause unnecessary overhead. Instead, a connection is explicitly negotiated beforehand between all devices in its route. This turns out to be a useful practice, allowing to implement additional (security) features with relative ease.

Security

We envision this network to be used for a wide variety of different applications ranging from public file transfer (think video archives and scientific datasets) to private conversations (for example between a doctor and a patient) to ones requiring anonymity (possibly between a journalist and a source). The desired balance between privacy and efficiency is in our opinion different for each of these cases. Large public non-controversial data could be transferred without encryption, but it would still be nice to be assured of its integrity. The contents of the conversation between a doctor and a patient should be available to nobody else, but the fact that somebody talks to their doctor is in many cases not particularly incriminating. For one who seeks to supply the public with a controversial story, it would however be on the safe side to hide the fact of talking to the journalist at all and the ability to stay pseudonymous may be as crucial.

The decision of what level of security to use is left to the device that starts a connection. The other device may reinitiate the connection if it finds a higher level of security necessary or disagrees on any other specified setting. Note that an unencrypted connection can be thought of as a connection encrypted using the identity function and is therefore not explicitly addressed.

Security by impact on performance

The following properties can be targeted with reasonably little effort by only the two communicating devices and in our opinion should be enabled by default:

- Confidentiality – the data sent over the network should be unreadable to anyone except the recipient.
- Integrity – the data is really from the device it claims to be and has not been tampered with by anyone else.
- Repudiability – the network should not enable any recipient to convince others that a packet it just received actually came from the sender.

- Forward secrecy – even someone who records the network traffic and later obtains the senders' and receivers' secret keys should be unable to decrypt it.
- The content of the packets should not disclose any information about identities of the sender and the receiver. Even the forwarders do not have to know where a packet originally came from or where it is going.

With additional work by packet forwarders and at the cost of speed, the effect of the following weaknesses can be reduced:

- Inspecting content of the packets that two devices send and receive can reveal if they are talking to each other. This can be prevented with the cost of some (symmetric) cryptographic steps by some of the forwarders, modifying the packets beyond recognition.
- Inspecting when two devices send and receive packets can reveal whether they are talking to each other or not. This can be mitigated by an artificial delay by some of the forwarders, obviously at the cost of increased latency.
- Physical location is usually impractical to hide. Assuming optimal routing, route latency can be used to estimate the location of any device. The most fundamental example of this comes from the limiting speed of light. If the time it takes to get a response from a device is $2 \cdot t = 2$ms then that device cannot be further away than $c \cdot t = 300$km. Additionally, a device's location can be estimated by comparing its latencies for communicating through different devices whose locations are known. Picking needlessly lengthy routes would hide the initiator's location from the other party.

Configurations for use cases

Data privacy (recommended defaults)

This configuration aims to provide confidentiality, integrity, repudiability and forward secrecy of the data sent over the network. Additionally, the content of the packets does not disclose information about the identities of the sender and the receiver. Each device in the route is only told about the next and previous hop. This does NOT prevent adversaries from deducing the used route by observing traffic on the devices or links in it – the packets can be recognized if re-encountered.

The initiator chooses a connection identifier, which will be the same for outgoing and incoming packets. An encrypted routing message is sent to each device on the route, asking them to forward packets with that identifier to the next or previous device based on from which they are received. In order not to reveal its identity to the forwarders, the initiator uses a disposable key for routing messages.

The initialization message is encrypted with the receivers public key and being able to decrypt it confirms the receivers identity.

A shared secret key is negotiated in a way that provides forward secrecy (such as CurveCP hello-cookie-auth sequence) and all further packets are encrypted using that.

In case the initator wishes to prove its identity to the receiver, they may vouch for the connection by sending the following.

- Their true public key
- A copy of the disposable public key encrypted and authenticated using its true public key

Anonymity

The goal of this configuration is to thwart some traffic analysis attacks while keeping the latency tolerable; in fact it can be seen as a port of The Onion Router[@tordesign] algorithm. The default configuration is used as a starting point. Some of the forwarders are asked to alter packets en route so that these cannot be recognised later.

With anonymizing forwarders, all routing messages are exchanged using a connection that provides forward secrecy. This is different from the default configuration and neccessary only to make it impossible for an adversary who has recorded the network traffic to later force each forwarder to reveal the next one.

Masking packets without causing overhead requires some tricks. Each anonymizing forwarder is asked to perform the following steps before sending a packet to the next device:

- Add or remove n bytes of random padding at the end of the packet.
- Decrypt the message body using its nonce and the specified key.
- Add or remove n bytes of random padding at the end of the packet.
- Apply a keyed pseudorandom permutation to the nonce.
- Delay a specified amount of time.
- Set the route identifier to the one for next hop.

The initiator specifies the following details for each forwarder for each direction:

- identifier for the incoming hop (the routing identifier)
- amount of padding to add or remove before decryption
- key to use for decrypting the message and rotating the nonce
- amount of padding to add or remove after decryption
- distribution of planned delays

- identifier for the outgoing hop

The question how much it is wise to delay or pad the packets at intermediate hops remains unsolved. The space reserved for padding added by anonymizing forwarders still counts towards the maximum transmission unit. The effect of delaying packet forwarding is obvious, but it's still hard to decide the appropriate amount of delay. Finding a suitable balance between useful content and obfuscating noise is not addressed here. Note that even without any additional delays or padding this configuration should provide a level of anonymity comparable to TOR.

An adversary who controls multiple (but not all) devices in the route cannot determine through which of the devices controlled by them the route goes by inspecting the packet contents. If not altered, timings and packet sizes can also be used to correlate traffic. This is extensively discussed in the TOR design paper[@tordesign] without reaching any conclusive solution. As a workaround, forwarders can be requested to add delays and add or remove padding.

The anonymizing forwarders cannot be forced to reveal the connection details after the connection is closed because the details are erased from memory together with the short-term key that was used to decrypt the request containing these details.

Such setup ultimately depends on the forwarders' co-operation – if every device in the route leaks the information available to it to the adversary, the route is compromised, be it intentional or not. To this end, all devices are required to either carry out all instructions exactly as requested or notify the requester of the failure or refusal to comply as early as possible (right after receiving the request message).

To prevent the creation of routing loops, packets with an identifier should be accepted from only one link. If a packet has a known routing identifier, but the route identified was configured using a different link, the packet must be discarded. Similarly a device that offers to change the routing identifier should to no link send packets that have the same routing identifier but were received from different links.

Availability of routing information

To connect to any other device as described above, a route to it and the public keys of all devices in the route have to be known. It is, however, infeasible for all devices to store and update information about all other devices and links. Instead, devices maintain information about the current state of some subset of the network. For every device stored, it is also necessary to store a route to it – if one still has to query others for a route to a device, they could get that device's public key as well with the same query. The part of the network a

device keeps track of should therefore consist only of other devices physically connected to that device.

To avoid the motivation to query for routes to devices kept track of (to check whether the known route is the best), devices should keep track of parts of the network closest to them. Defining the distance to a link as the minimum of distances to its endpoints, it can be stated that possessing all information about the network details no further away than some distance, optimal routes to all devices closer than that distance can be determined without any queries to other devices.

Simply tracking a constant number of devices and/or links to which the reported routes are the shortest by itself is susceptible to attack by announcing many link descriptions promising efficient connection to fake or useless devices. This problem can be thought of as one of device prioritization and a possible solution is to prefer devices which are supported by a valid chain of trust. Using a trust metric in addition to distance is also useful for route finding because keeping track of more trusted devices means more information about reliable routes.

Changes in networks

Real-life networks are not completely static, even when expected or desired to be. The changes are relatively rare and clearly outnumbered by packets. If there was a network where every device relied on storing accurate information about all others, it would be necessary to flood information about changes through the whole network. By storing information about the network structure in a local manner, the flooding can be limited to devices close to the change.

Every device that causes or detects a change in the network structure informs other devices of it by broadcasting a respective routing message. After receiving such a message the receiver decides whether the described change should be incorporated into its (partial) map of the network. As all devices keep an accurate map only about a local subset of the network, the impact of changes is also local. A change that a device adds to its map is also likely to be interesting to its neighbors. The converse is also true for the fundamental "k nearest devices" map if all devices use the same k. Therefore, a simplified rule for informing other devices about relevant changes in the network can be stated as follows: every device has to forward all changes it incorporates into its map.

There fortunately is no need to artificially limit the mapping strategies that other devices can use by only forwarding information about changes that would be interesting to them if they used the same mapping scheme. Instead, changes interesting to us should be given a priority, but during low load every change should be forwarded. The freedom to choose the mapping heuristic is important because actual links are not described by "length", but with multiple characteristics such as round trip time, maximal transmission unit and packet loss rate and devices can disagree on relative importance of these.

Bootstrapping

When a new device joins a network for the first time or moves, they have to obtain up to date information about the neighborhood. Receiving a copy of a neighbor's map would provide necessary information. The device sharing a map should take care of their privacy, primarily by omitting some devices from the response. For this to be possible, it is necessary to keep the routes chosen to be included with respect to public rules separate from the usage-based cache, if it is used.

Heuristics

There are two main points in this scheme that require clever choices to be made by computers. The simpler is the question of route selection: given some route descriptions that lead to a destination, determine the one most suitable for the intended use. We find the task of finding the routes in the first place significantly harder to tackle. The main design goal of this routing scheme is to allow different kinds of heuristics to be used interchangeably without compromising confidentiality of the data or requiring compatibility from all devices. *This is an area for further research.*

Route selection

Input consists of some routes and optionally a description of the desired usage. Each link in each route is labeled with some properties like estimated round trip time, minimal transmission unit, time of issue or packet loss rate. The best route should be **output**. Choosing the data to label the links with can be considered the **auxiliary** part of this task.

Route finding

Input is the identifier (public key hash) of the device to whom a route is required. The **output** should be one or multiple routes to that device and the routes should be as good as possible. Unlike during route selection, queries to other devices can be made. **Auxiliary** tasks include constructing and distributing routing information in a way that it is systematically and globally accessible.

References