Using the HiiMap Mapping System as a Public Key Infrastructure

Oliver Hanka
Technische Universität München
Institute of Communication Networks
80333 Munich, Germany
oliver.hanka@tum.de

I. Introduction

The HiiMap architecture [1] is a Next Generation Network concept which is based around the locator/identifier-split idea. As with each NGN architecture, security was a major concern from the beginning and one of the design goals was, to integrate the functionality to establish secure connections into the architecture. The 2-tier hierarchical structure of the HiiMap mapping system—a Global Authority as single point of trust and the mapping databases divided into regions—builds the foundation to support security in the architecture.

Functionality like confidentiality, integrity and authenticity can be realized by the public/private key principle. This mechanism, however, needs some form of key distribution. This short abstract describes, how a public key infrastructure can be integrated into the HiiMap mapping system.

II. Loose vs. tight identifier–key binding

The *Host Identity Protocol (HIP)* by Moskowitz et al. [2] uses the public key as the Host Identity. To deal with different key length, the Host Identity is hashed to a 128-bit value. This value is called the Host Identity Tag and used as the identifier. This has the benefit that any peer can verify whether the supplied public key is the correct one for a specific node or not. This can be done without the need of an additional public key infrastructure.

This approach, however, has four disadvantages:

- In case the key or identifier changes, the other one has to as well
- No possibility to withdraw a key
- Additional trust entity required to verify the relationship between the identifier and a legal–person
- Random key–pair guess

We, therefore, suggest to only loosely couple the public key with the identifier. This means that there is no mathematical relationship between those two values. We propose to use the mapping system as a public key infrastructure and store the public key along with the locator in the mapping system.

III. PKI and the mapping

In today's Internet, the public key infrastructure is separated from all network services. This means that additional resources for the PKI must be provided despite all the network elements already in place for other functionality (e.g. DNS server). Contrary to this, we propose to integrate the PKI into the mapping system for the HiiMap [1] architecture. This has the benefit that resources can be shared between functionalities and maintenance can be kept significantly lower compared to operating separate services.

Each mapping entry consist of an identifier as the primary key and a set of locators by which the node currently can be reached. To combine the PKI with the mapping system, only the public key of each node must be additionally stored for each mapping entry. This means that no additional protocol or infrastructure must be provided for querying and storing the public keys. Because the public key is a very static value and not expected to change frequently, the additional burden for the mapping system is limited and the public key databases can be optimized for frequent read requests—contrary to frequent read and write requests for the locators.

In comparison with today's public key infrastructure, it is also not necessary to provide additional lists for key revocation. This is implicitly realized by the loose identifier–public key binding. Whenever a public key for a certain identifier changes, the old public key implicitly becomes invalid.

A. Trusting the mapping system

By storing the public key at only one location (region) in the mapping system, however, the user heavily depends on the trustworthiness of that particular location. In case the mapping service provider collaborates with an attacker, it could send a wrong or manipulated public key to the client. Any security functionality based on the public/private key principal, therefore, would be rendered useless. Even worse, the client considers the connection to be secure while in fact talking directly to the attacker.

Therefore, we propose to distribute several copies of the public key to various independent locations (regions) in the mapping system. Figure 1 illustrates the basic principal.

The client first queries the responsible region (RR) of the identifier it wants to communicate with. As response, the RR replies with the locator and public key stored for that identifier. In a next step, the client queries additional regions for the public key. We will explain which regions to query in the next section. After receiving all requested public keys, the client compares these. In case they do match, it is very likely that the public key is the correct one. Contrary, if they differ, the client can either stop the

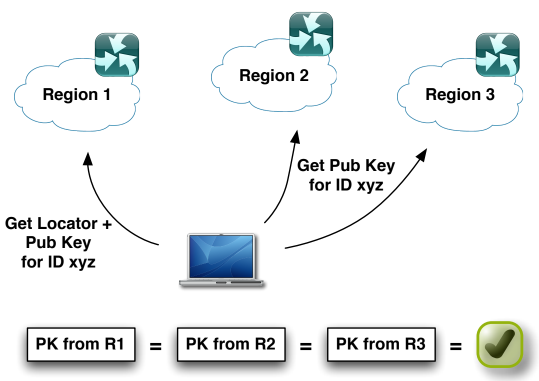


Figure 1. The public key is stored at multiple regions

communication setup or decide which is the correct key based on the majority principle.

There is one special case, however. If the public key from the RR differs from the other ones, then the retrieved locator must be considered incorrect as well. This is because having identified the RR as accessory or even the attacker itself, it is very likely that the locator has been modified as well and is now pointing directly towards the attacker.

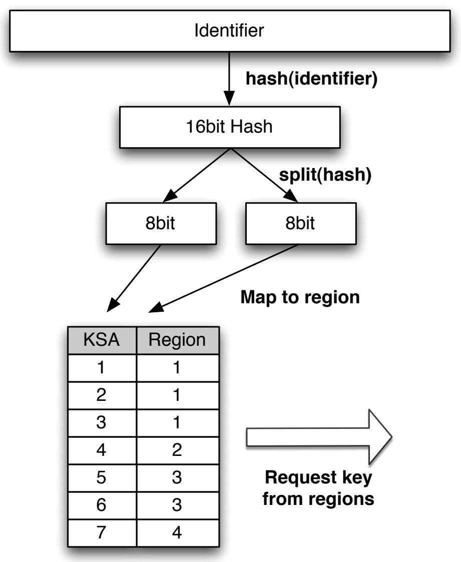
A solution to this problem would be to also replicate the locator over several other regions. This, however, is not a good idea performance wise. The locator is the entry in the mapping system which will be updated and changed frequently. In case several regions hold a copy of it, these changes have to be carried out to all of them. The public key on the other hand is expected to change very rarely and thus causing very little update traffic.

B. Determining the storage location

Having copies of the public key distributed over several locations in the mapping system, one questions remains: In which way does the client learn about the storage location of the additional copies?

Storing the list of the additional locations at the RR does not solve the problem. In case the RR is the attacker, it can simply manipulate this list as well and distribute the malicious key to collaborating regions. Therefore, the client must learn the information about the storage locations in another way.

For the following proposal, we assume that less than 256 regions exist and the key is distributed to two additional regions. After having received the locator and public key from the RR, the client hashes the identifier to an 16-bit value. The 16-bit value is split into two halves (8-bit each). Each 8-bit value represents the storage location of one of the public key copies. We will call it key storage address space (KSA) from now on. Since the 8-bit address space for the regions is not completely full, a mapping directive is required. This mapping directive can be downloaded from the global authority. It is sufficient to do this very seldom, as the directive is expected to change very rarely. For each value in the KSA, the mapping directive specifies a region, where the key is stored. This means, that a single region can be responsible for several KSA values. In that way, the load can be fairly distributed over all regions depending on their size. Figure 2 illustrates the process.



KSA	Region
1	1
2	1
3	1
4	2
5	3
6	3
7	4

Figure 2. Retrieving the additional key storage locations

Should the hashing and mapping to regions result in two copies of a key being stored at the same region, the first copy is stored at this region and the second copy at the region with the next higher region number.

C. Initial key upload

Whenever a new node connects to the network for the first time, its public key must be stored in the mapping system in a secure way. For security and usability reasons, we propose to use cryptographic smart cards to store the public-, private key and identifier of any node at the users side [3]. The smart card is issued by the global authority—or one of its delegates—and initial generates a public–private key pair on chip. Before shipping the smart card to the user, the public key is derived from it and stored in the mapping system. In that way, the global authority is never in possession of the private key and the public one is copied to the mapping server within a secure enviroment. For further details of this initial bootstrap see [3].

References

[1] O. Hanka, G. Kunzmann, C. Spleiß, J. Eberspächer, and A. Bauer, "*HiiMap: Hierarchical Internet Mapping Architecture*," *In First International Conference on Future Information Networks, Beijing, China, P.R. China*, pp. 17–24, Oct. 2009.

[2] R. Moskowitz and P. Nikander, "Host Identity Protocol," IETF, RFC 4423, May 2006.

[3] W. Fritz and O. Hanka, "Smart card based security in locator/identifier-split architectures," *International Conference on Networking*, vol. 0, pp. 194–200, 2010.