Garantir

# GaraSign Architecture

Version 1.21.0

## Table of Contents

# Preface

## Document Information

| | |
|---|---|
| Title | GaraSign Architecture |
| Product Version | 1.22.0 |
| Release Date | December 2022 |

## Trademarks

## Disclaimer

We have attempted to make these documents complete, accurate, and useful, but we cannot guarantee them to be without error or otherwise perfect. When we discover errors or omissions, or they are brought to our attention, we endeavor to correct them in succeeding releases of the product.

Please send any constructive comments on the contents of this document to the following email address: support@garantir.io

# Document Overview

GaraSign is made up of many components, some required and some optional. This document provides an overview of GaraSign's architecture and how these components integrate with each other.

## Intended Audience

All products produced by Garantir are designed to be installed, configured, operated, and maintained by personnel with the necessary knowledge, skill, training, and qualifications to safely perform their duties. This document is intended for personnel responsible for architecting, engineering, installing, configuring, operating, and/or troubleshooting enterprise signing solutions. It is assumed that the readers of this document and the users of its content are proficient with:

- Basic networking concepts
- Security concepts including, but not limited to, authentication, authorization, digital signatures, and logging

Additionally, it is strongly recommended that readers of this document first read the GaraSign datasheet.

## GaraSign Overview

GaraSign is a flexible enterprise cryptographic signing platform that carefully balances security and performance. Unlike most solutions today that require clients to upload the data to be signed to a central location, GaraSign is designed as a remote signing platform. As shown in Figure 1 below, GaraSign clients first hash the data to sign and then send the computed hash value over a secure channel to the GaraSign server for signing. Using this approach, no matter how large the data is, the data sent over the network is minimal which allows for optimum performance. Behind the GaraSign server sits one or more cryptographic tokens (e.g., HSMs, key managers, etc.) that GaraSign can offload signature processing to once the appropriate authentication and authorization checks have been made. By placing the GaraSign server between the client and the back-end HSM, not only is performance significantly increased, other enterprise features such as advanced authentication, enterprise logging, email notifications, and more are enabled. As an additional benefit, the GaraSign server acts as a buffer between your end clients and your HSMs, which not only significantly reduces the integration complexities that your clients must deal with but also helps to shield your cryptographic keys from attack and misuse.

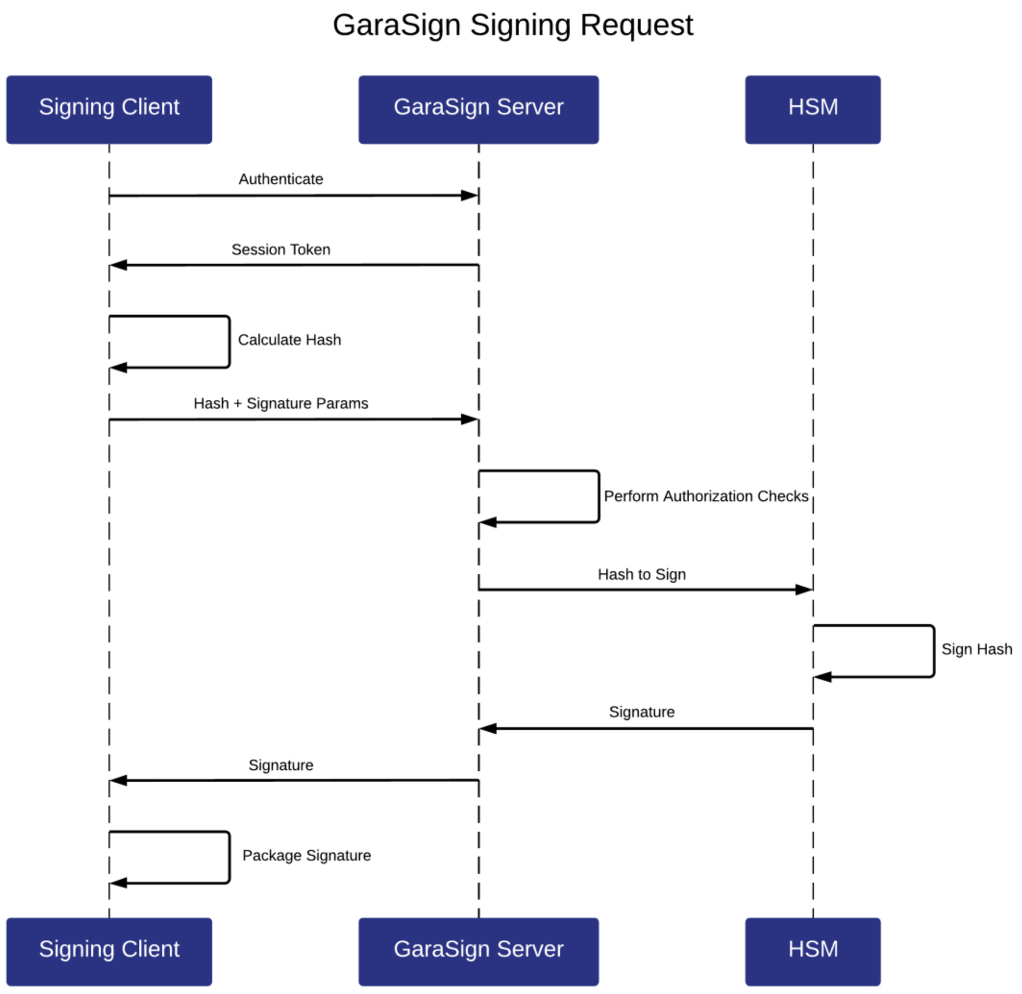

*Figure 1 - GaraSign Signing Request*

Note: while this document makes mention of REST servers, GaraSign is designed to sit on-premise in your network and/or in your virtual private cloud. Garantir does **not** have access to any of your infrastructure and does **not** control your cryptographic keys.

## Design Decisions

GaraSign was designed with the following principles in mind:

1. Minimal Network Usage
2. Security
3. Tunable Security & Performance
4. Open API
5. Scalable
6. Extendable
7. Work Anywhere
8. Easy to Integrate

### Minimal Network Usage

One of the biggest factors in performance degradation is network usage. To overcome this, GaraSign makes the minimal number of network calls and sends the least amount of data per network call as possible. While this is most apparent in the "local hash, remote sign" architecture, GaraSign also implements several other (configurable) network-saving features including client-side caching, server-side caching, and more.

### Security

Of course, any good signing solution must be secure. Garantir has gone to great lengths to ensure that every facet of GaraSign is secure. Please read the GaraSign Threat Model document for a detailed write up on the security of GaraSign.

### Tunable Security & Performance

Different customers have different security and performance requirements and many customers have different needs depending on the environment within their enterprise for which they are signing data. GaraSign accounts for this by allowing both high- and low-level configuration values that change the behavior of the system. For example, hash validation may be set on a per-key basis, notifications may be set on a per-key or per-user basis, clients can be configured to cache session tokens and certificates, etc.

### Open API

Both the signing and administration interfaces have open APIs that customers can choose to integrate with if they would like. This is useful for customers that have existing infrastructure they would like to integrate with or custom signing clients that GaraSign does not yet support.

### Scalable

All server-side nodes are stateless and designed to be run in high availability clusters or as single nodes. Customers may choose to add more nodes during peak time and shut them down when they are not needed (e.g., auto-scaling).

### Extendable
All logic implemented server-side has a defined interface. Customers with unique requirements to extend the functionality of GaraSign may implement the desired interfaces and plug them in at runtime without affecting the core GaraSign product.

### Work Anywhere
GaraSign is designed to run on a very simple platform that can be run on-premise in your data center, in the cloud, or in a hybrid environment between the two. This allows for maximum flexibility when architecting for high availability and disaster recovery.

### Easy to Integrate
GaraSign is designed to be simple to deploy and integrate with your environment. Furthermore, it is designed to integrate with existing signing tools and enterprise systems.

## Architecture
As shown in Figure 2 below, the entire GaraSign architecture has many components. To make this architecture easier to understand, this document takes a "zooming in" approach where each logical coupling of components is looked at from the viewpoint of the users and administrators who interact with them.

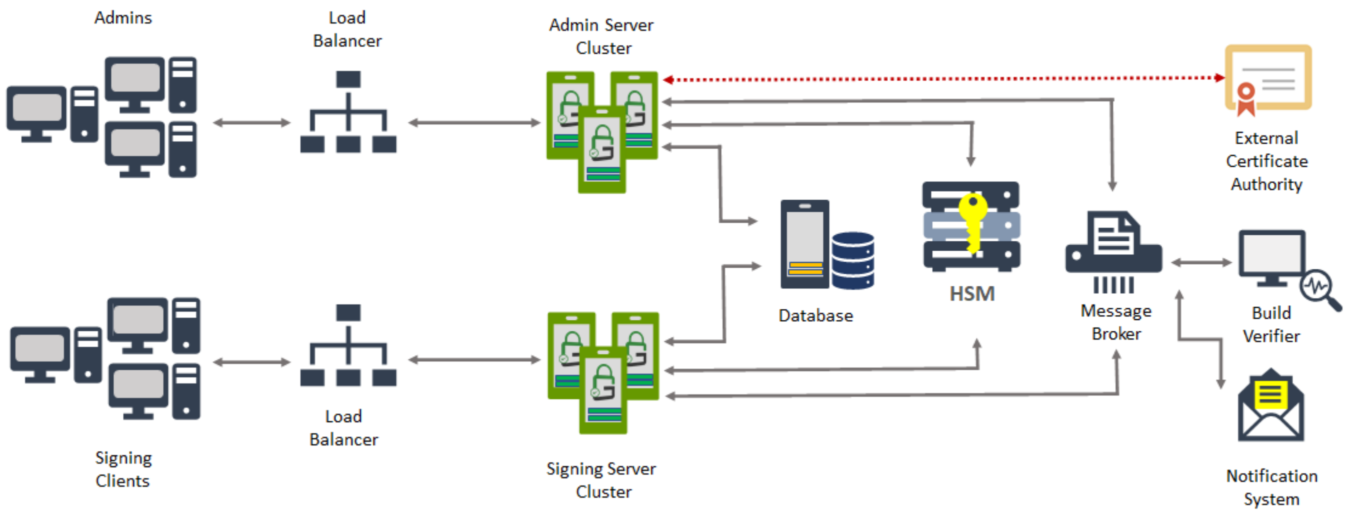

*Figure 2 - GaraSign Architecture Diagram*

GaraSign is made up of the following components:

1. Cryptographic Token – The cryptographic device(s) that store the signing keys (usually one or more HSMs)
2. GaraSign Signing Server – The stateless REST server that sits in front of the cryptographic tokens that store the signing keys
3. GaraSign Signing Clients – Clients that allow the signing tools they integrate with to locally hash data and offload signature generation to the GaraSign Signing Server
4. GaraSign Administration Server – The stateless REST server that allows administrators to perform their duties (e.g., user management, key management, etc.)

5. GaraSign Administration Clients – The client software that allows administrators to interface with the Administration Servers to perform administrative duties
6. GaraSign Hash Validator – The servers configured to validate hash values either before or after signing occurs
7. GaraSign Notification Server – The server that sends out notifications, usually in the form of email
8. GaraSign Message Broker – The messaging system that allows the Signing, Administration, Hash Validator, and Notification servers to communicate
9. Database – The database that stores persisted information such as users, key metadata, and signature history
10. Load Balancer(s) – Used to balance load in front of the various service endpoints

## Cryptographic Token

The cryptographic device that stores the private keys and certificates used for signing. In most deployments these tokens will be hardware security modules (HSMs), but GaraSign supports any cryptographic token that has a programmatic interface (e.g., key manager, software keystore, etc.). GaraSign supports multiple different cryptographic tokens simultaneously, even if those tokens are provided by different vendors.

## GaraSign Signing Server

The GaraSign Signing Server is the component that gets the most use in the entire architecture. Its main purposes are to authenticate clients, give clients access to their keystore objects (i.e., certificate chains and key identifiers), and facilitate signing requests between the client and the backend cryptographic token(s). Designed as a stateless REST server, customers can choose to stand up one signing server or multiple servers in a high availability cluster. Ignoring features like administration, notifications, and hash verification, the core of GaraSign looks like Figure 3 below.

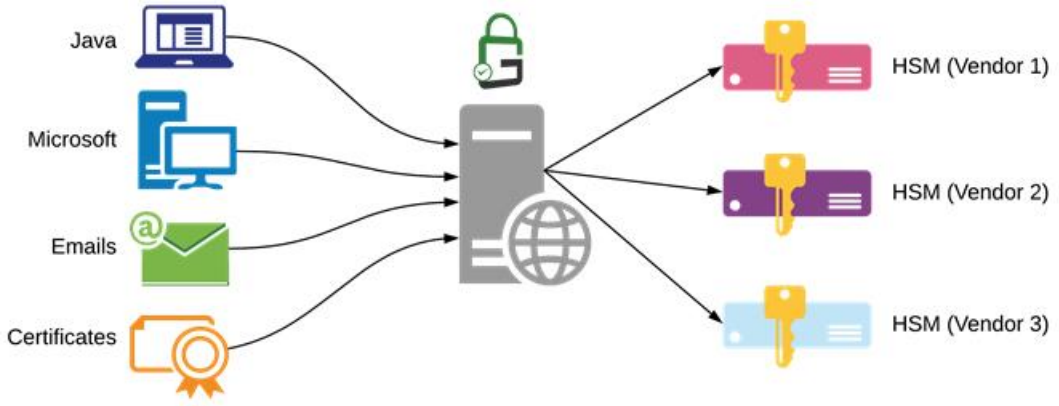

*Figure 3 - Signer's Perspective*

Note: Figure 3 only shows a small subset of the supported clients. GaraSign supports many other integrations including codesign on macOS, RPM, PKCS#7 (i.e., CMS), and many more.

## GaraSign Signing Clients

The GaraSign Signing Clients are the software components that hash data and interact with the GaraSign Signing Server to perform authentication and sign the hashes that they compute. Wherever possible, these clients are implemented as software providers that existing signing tools can use to offload cryptographic processing. In certain situations where the existing signing tools do not support a pluggable cryptographic software provider, Garantir provides a replica of the signing tool that will interface with the GaraSign Signing Server for signature generation. While GaraSign comes with its own set of signing clients, some customers may choose to make use of the open signing and authentication REST APIs to create their own custom signing clients.

## GaraSign Administration Server

The GaraSign Administration Server allows administrators to perform administrative duties such as user management, key management, group management, and more via a GUI or command line interface. Designed as a stateless REST server, customers can choose to stand up one administration server or multiple ones in a high availability cluster. To keep a strong separation between operational and administrative use, these servers are designed to reside on different hosts than the Signing Servers.

## GaraSign Administration Clients

The GaraSign Administration Clients allow administrators to interface with the Administration Servers to perform their duties. GaraSign comes with its own administration client but some clients may wish to make use of the open REST API to create their own administration clients.

## GaraSign Hash Validator

The Hash Validators are an optional component of the architecture that allow the signing servers to validate the hash to be signed either before or after signing occurs. These validators can be configured to validate several things including, but not limited to, the hash to sign, static code analysis, automated testing, code review results, etc. When run in pre-validation mode, the signing server will **not** sign the hash until hash validation has completed successfully, as shown in Figure 4 below. This is useful for highly sensitive keys where security requirements are the strictest.

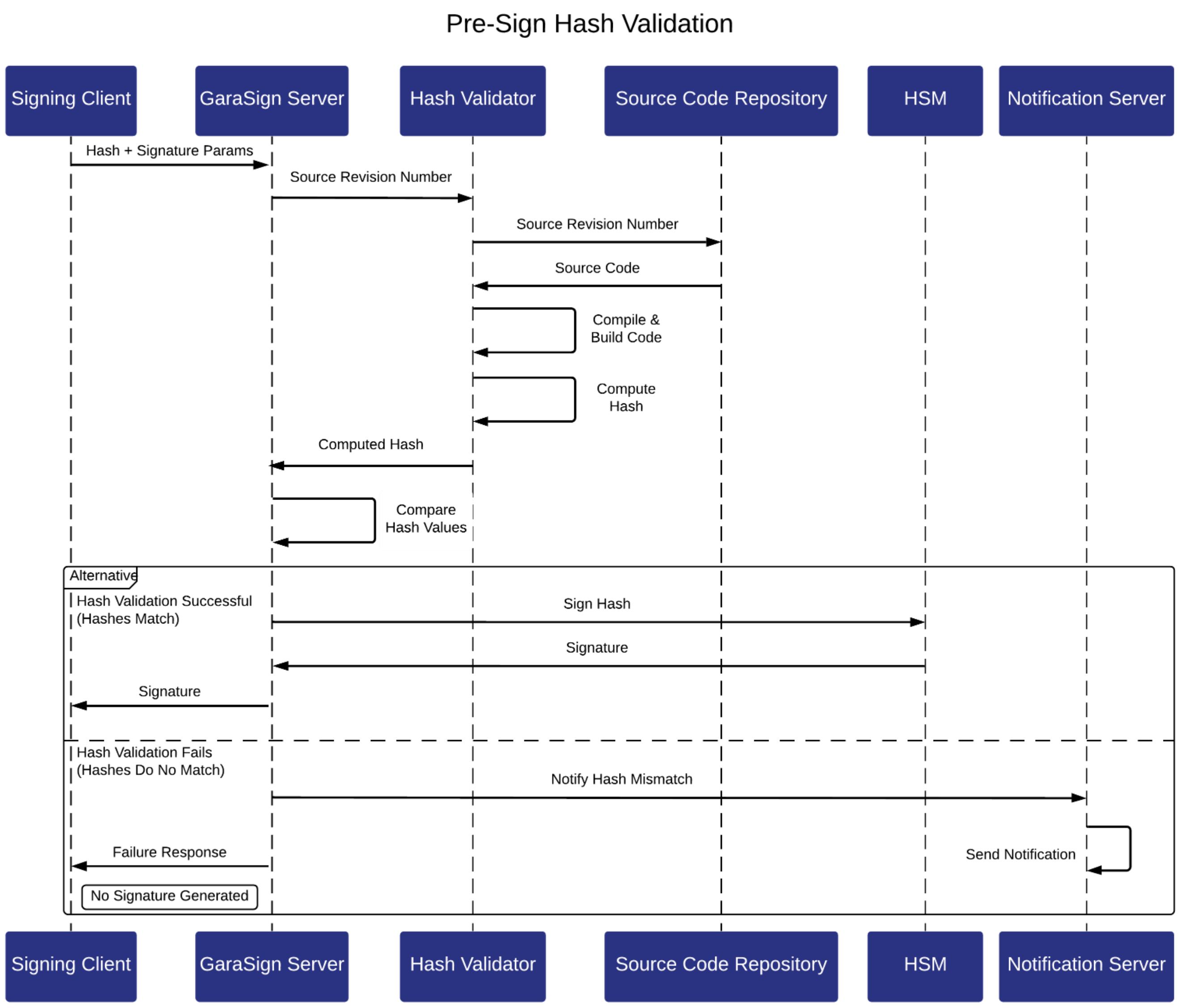

*Figure 4 - Pre-Hash Validation*

When run in post-validation mode, the signing server signs the data *before* validation completes and sends a notification to the configured parties if validation fails, as shown in Figure 5 below. This is useful for less sensitive keys and in environments where performance matters more than security.

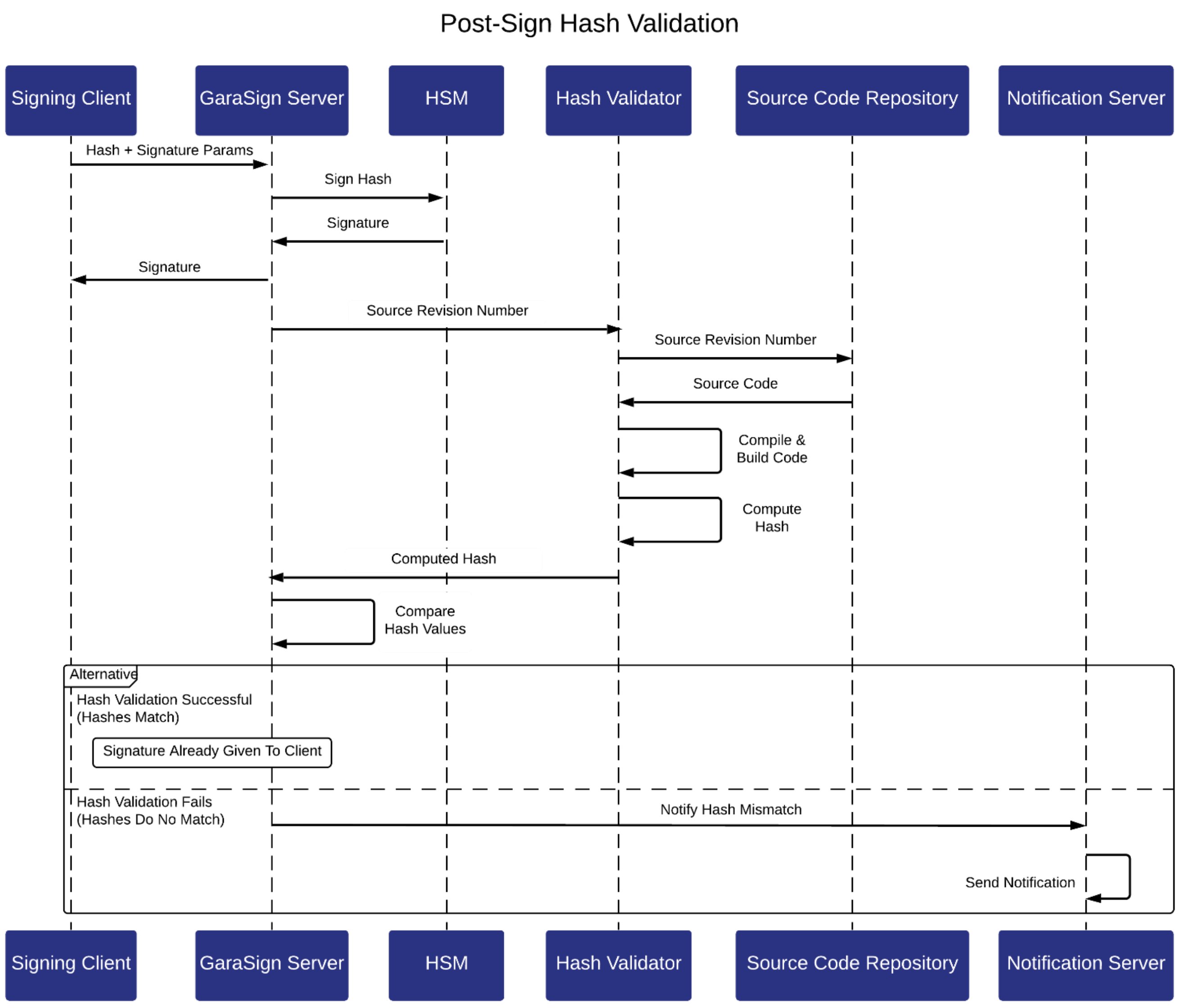

*Figure 5 - Post-Hash Validation*

Please see the GaraSign Threat Model document for more information on how hash validation can be configured in your environment to optimally balance security and performance.

## GaraSign Notification Server

The GaraSign Notification Server sends notifications to users and administrators about important system events. On the signing side, the notification server notifies users that signature requests have completed or failed. On the administrative side, the server notifies administrators of auditable actions. By default, the server is configured to send email and/or Slack notifications, but it can be configured to integrate with a wide variety of notification methods.

### GaraSign Message Broker
The GaraSign Message Broker allows the server-side components of GaraSign to communicate with each other. All communication via the message broker is done over TLS and all components must first authenticate themselves to the message broker. Please see the GaraSign Threat Model document for more information on the security of the message broker.

### Database
The database stores the persistent data that is needed by the administration and signing servers. GaraSign comes with a default database implementation but customers with unique requirements (e.g., integration into a legacy signing environment) may choose to implement the GaraSign Database Interface and use their own custom database.

### Load Balancer
All server-side components of GaraSign are designed to work with or without a load balancer. GaraSign does **not** provide its own load balancer but will work with any commercially available load balancer provided that it supports the *X-Forwarded-For* HTTP header.

## Frequently Asked Questions

### Do the clients ever get access to the signing keys?
No. The signing keys always stay within their cryptographic tokens. In most cases this means that the signing keys always reside within an HSM.

### If only the hash of the data is signed, how do you ensure that rogue data isn't being signed?
This is covered in more detail in the GaraSign Threat Model document, but the general idea is that only authenticated clients may sign data with keys they are authorized to use, and customers may choose to validate hashes prior to signing. GaraSign supports two methods of hash validation – automated and manual. For more information on automated hash validation, please see the GaraSign Hash Validator section. For more information on manual hash validation, please see the GaraSign Threat Model document.

### We have a mix of high and low security keys in the enterprise. How do we balance performance and security?
Policy can be set on a per-user or per-key basis. For example, customers may choose to set key-1 to not require any validation, key-2 to require post-hash validation, key-3 to require pre-hash validation, and key-4 to require pre-hash validation and manual validation. Additionally, customers can choose to create different users for different keys and use different authentication methods for each of those users.

### My company has multiple different HSMs in the enterprise. Do we need a separate GaraSign instance for each?
While that approach would work, no, you do not need a separate instance for each. GaraSign is designed to sit in front of multiple different types of cryptographic tokens simultaneously.

### We are in the process of migrating to the cloud. How can we deploy GaraSign so it works today but will also work in our future cloud environment?
GaraSign is designed to run on-premise or in the cloud and can even run in a hybrid environment. As long as there is network communication between the nodes, the system will work. GaraSign even supports the HSMs

and key managers offered by certain cloud providers to allow you to use hybrid or fully cloud-based cryptographic tokens. For customers who wish to keep their cryptographic tokens on premise but utilize the cloud for scaling, GaraSign servers in the cloud can make use of the cryptographic tokens on premise as long as the network connection is available.