Google Cloud

Encryption in Transit in Google Cloud

December, 2017

Table of contents

This is the third whitepaper on how Google uses encryption to protect your data. In this whitepaper, you will find more detail on encryption in transit for Google Cloud, including Google Cloud Platform and Google Workspace.

For all Google products, we strive to keep customer data highly protected and to be as transparent as possible about how we secure it.

The content contained herein is correct as of December 2017. This whitepaper represents the status quo as of the time it was written. Google Cloud's security policies and systems might change going forward, as we continually improve protection for our customers.

CIO-level summary

- Google employs several security measures to help ensure the authenticity, integrity, and privacy of data in transit.
- For the use cases discussed in this whitepaper, Google encrypts and authenticates data in transit at one or more network layers when data moves outside physical boundaries not controlled by Google or on behalf of Google. All VM-to-VM traffic within a VPC network and peered VPC networks is encrypted.
- Depending on the connection that is being made, Google applies default protections to data in transit. For example, we secure communications between the user and the Google Front End (GFE) using TLS.
- Google Cloud customers with additional requirements for encryption of data over WAN can choose to implement further protections for data as it moves from a user to an application, or virtual machine to virtual machine. These protections include IPSec tunnels, Gmail S/MIME, managed SSL certificates, and Istio.
- Google works actively with the industry to help bring encryption in transit to everyone, everywhere. We have several open-source projects that encourage the use of encryption in transit and data security on the Internet at large including Certificate Transparency, Chrome APIs, and secure SMTP.
- Google plans to remain the industry leader in encryption in transit. To this end, we dedicate resources toward the development and improvement of encryption technology. Our work in this area includes innovations in the areas of Key Transparency and post-quantum cryptography.

Introduction

Security is often a deciding factor when choosing a public cloud provider. At Google, security is of the utmost importance. We work tirelessly to protect your data—whether it is traveling over the Internet, moving within Google's infrastructure, or stored on our servers.

Central to Google's security strategy are authentication, integrity, and encryption, for both data at rest and in transit. This paper describes our approach to encryption in transit for Google Cloud.

For data at rest, see Encryption at Rest in Google Cloud Platform. For an overview across all of Google Security, see Google Infrastructure Security Design Overview.

Audience: this document is aimed at CISOs and security operations teams using or considering Google Cloud.

Prerequisites: in addition to this introduction, we assume a basic understanding of encryption and cryptographic primitives.

Authentication, Integrity, and Encryption

Google employs several security measures to help ensure the authenticity, integrity, and privacy of data in transit.

- **Authentication**: we verify the data source, either a human or a process, and destination.
- **Integrity**: we make sure data you send arrives at its destination unaltered.
- **Encryption**: we make your data unintelligible while in transit to keep it private. Encryption is the process through which legible data (plaintext) is made illegible (ciphertext) with the goal of ensuring the plaintext is only accessible by parties authorized by the owner of the data. The algorithms used in the encryption process are public, but the key required for decrypting the ciphertext is private. Encryption in transit often uses asymmetric key exchange, such as elliptic-curve-based Diffie-Hellman, to establish a shared symmetric key that is used for data encryption. For more information on encryption, see Introduction to Modern Cryptography.

Encryption can be used to protect data in three states:

- **Encryption at rest** protects your data from a system compromise or data exfiltration by encrypting data while stored. The Advanced Encryption Standard (AES) is often used to encrypt data at rest.

- **Encryption in transit**: protects your data if communications are intercepted while data moves between your site and the cloud provider or between two services. This protection is achieved by encrypting the data before transmission; authenticating the endpoints; and decrypting and verifying the data on arrival. For example, Transport Layer Security (TLS) is often used to encrypt data in transit for transport security, and Secure/Multipurpose Internet Mail Extensions (S/MIME) is used often for email message security.
- **Encryption in use**: protects your data in memory from compromise or data exfiltration by encrypting data while being processed, e.g. Confidential Computing.

Encryption is one component of a broader security strategy. Encryption in transit defends your data, after a connection is established and authenticated, against potential attackers by:

- Removing the need to trust the lower layers of the network which are commonly provided by third parties
- Reducing the potential attack surface
- Preventing attackers from accessing data if communications are intercepted

With adequate authentication, integrity, and encryption, data that travels between users, devices, or processes can be protected in a hostile environment. The remainder of this paper explains Google's approach to the encryption of data in transit and where it is applied.

Google's Network Infrastructure

Physical boundaries

Google applies different protections to data in transit when it is transmitted outside a physical boundary controlled by or on behalf of Google. A physical boundary is the barrier to a physical space that is controlled by or on behalf of Google, where we can ensure that rigorous security measures are in place. Physical access to these locations is restricted and heavily monitored. Only a small percentage of Google employees have access to hardware. Data in transit within these physical boundaries is generally authenticated, but may not be encrypted by default - you can choose which additional security measures to apply based on your threat model.

Due to the scale of the global Internet, we cannot put these same physical security controls in place for the fiber links in our WAN, or anywhere outside of physical boundaries controlled by or on behalf of Google. For this reason, we automatically enforce additional protections outside of our physical trust boundary. These protections include encryption of data in transit.

How traffic gets routed

To fully understand how encryption in transit works at Google, it is also necessary to explain how traffic gets routed through the Internet. This section describes how requests get from an end user to the appropriate Google Cloud service or customer application, and how traffic is routed between services.

A **Google Cloud service** is a modular cloud service that we offer to our customers. These services include computing, data storage, data analytics and machine learning. For example, Google Cloud Storage and Gmail are both Google Cloud services. A **customer application** is an application hosted on Google Cloud that you, as a Google customer, can build and deploy using Google Cloud services. Customer applications or partner solutions that are hosted on Google Cloud are not considered Google Cloud services[1]. For example, an application you build using Google App Engine, Google Kubernetes Engine, or a VM in Google Compute Engine is a customer application.

The five kinds of routing requests discussed below are shown in Figure 1. This figure shows the interactions between the various network components and the security in place for each connection.

End user (Internet) to a Google Cloud Service

Google Cloud services accept requests from around the world using a globally distributed system called the Google Front End (GFE). GFE terminates traffic for incoming HTTP(S), TCP and TLS proxy traffic, provides DDoS attack countermeasures, and routes and load balances traffic to the Google Cloud services themselves. There are GFE points of presence around the globe with routes advertised via unicast or Anycast.

GFEs proxy traffic to Google Cloud services. GFEs route the user's request over our network backbone to a Google Cloud service. This connection is authenticated and encrypted from GFE to the front-end of the Google Cloud service or customer application, when those communications leave a physical boundary controlled by Google or on behalf of Google. Figure 1 shows this interaction (labelled connection A).

End user (Internet) to a customer application hosted on Google Cloud

There are several ways traffic from the Internet can be routed to a customer application you host on Google Cloud. The way your traffic is routed depends on your configuration, as explained below. Figure 1 shows this interaction (labelled connection B).

Cryptographic protections are provided to example configurations as follows:

- If you are connecting via the VM's external IP, or via a network-load-balanced IP, the connection does not go through the GFE. This connection is not encrypted by default and its security is provided at the user's discretion.
- If you are connecting from a host on your premises to a Google Cloud VM via a Cloud VPN, the connection goes from/to your on-premises host, to the on-premises VPN, to the Cloud VPN, to the Google Cloud VM. The connection is protected from the on-premises VPN to the Cloud VPN with IPSec. The connection from the Cloud VPN to the Google Cloud VM is authenticated and encrypted by Google.
- If you are connecting via Cloud Dedicated Interconnect, the connection goes from/to your on-premises host directly and the connection does not go through the GFE. This connection is not encrypted by default and its security is provided at the user's discretion. You can use the Transport Layer Security (TLS) Layer 7 cryptographic protocol to encrypt application traffic over Dedicated Interconnect.
- If you are using a Google Cloud HTTP(S) or TCP/SSL proxy Load Balancer external load balancer, consult Load Balancer product documentation.

Virtual Machine to Virtual Machine

VM-to-VM connections within VPC networks and peered VPC networks inside of Google's production network are authenticated and encrypted. This includes connections between customer VMs and between customer and Google-managed VMs such as Cloud SQL. Figure 1 shows this interaction (labelled connection C).

Connectivity to Google APIs and services

Traffic handling differs depending on the location of the Google Cloud service:

- Most Google APIs and services are hosted on Google Front Ends (GFEs); however, some services are hosted on Google-managed instances. For example, private services access and GKE masters for private clusters are hosted on Google-managed instances.

  With Private Google Access, VMs that don't have external IP addresses can access supported Google APIs and services, including customer applications hosted on App Engine. For more information about access to Google APIs and services, see Private access options for services.

- If a Compute Engine VM instance connects to the external IP address of another Compute Engine VM instance, traffic remains in Google's production network. Systems that are outside of Google's production network that connect to an external IP address of a Compute Engine VM instance have traffic routed over the internet.

Figure 1 shows an external path (labeled connection D). Typical cases of this kind of routing request are:

- From a Compute Engine VM to Google Cloud Storage
- From a Compute Engine VM to a Machine Learning API

From the VM to the GFE, Google Cloud services support protecting these connections with TLS by default[2]. The connection is authenticated from the GFE to the service and encrypted if the connection leaves a physical boundary.

Google Cloud service to Google Cloud service

Routing from one production service to another takes place on our network backbone and may require routing traffic outside of physical boundaries controlled by or on behalf of Google. Figure 1 shows this interaction (labelled connection E). An example of this kind of traffic is a Google Cloud Storage event triggering Google Cloud Functions. Connections between production services are encrypted if they leave a physical boundary, and authenticated within the physical boundary.

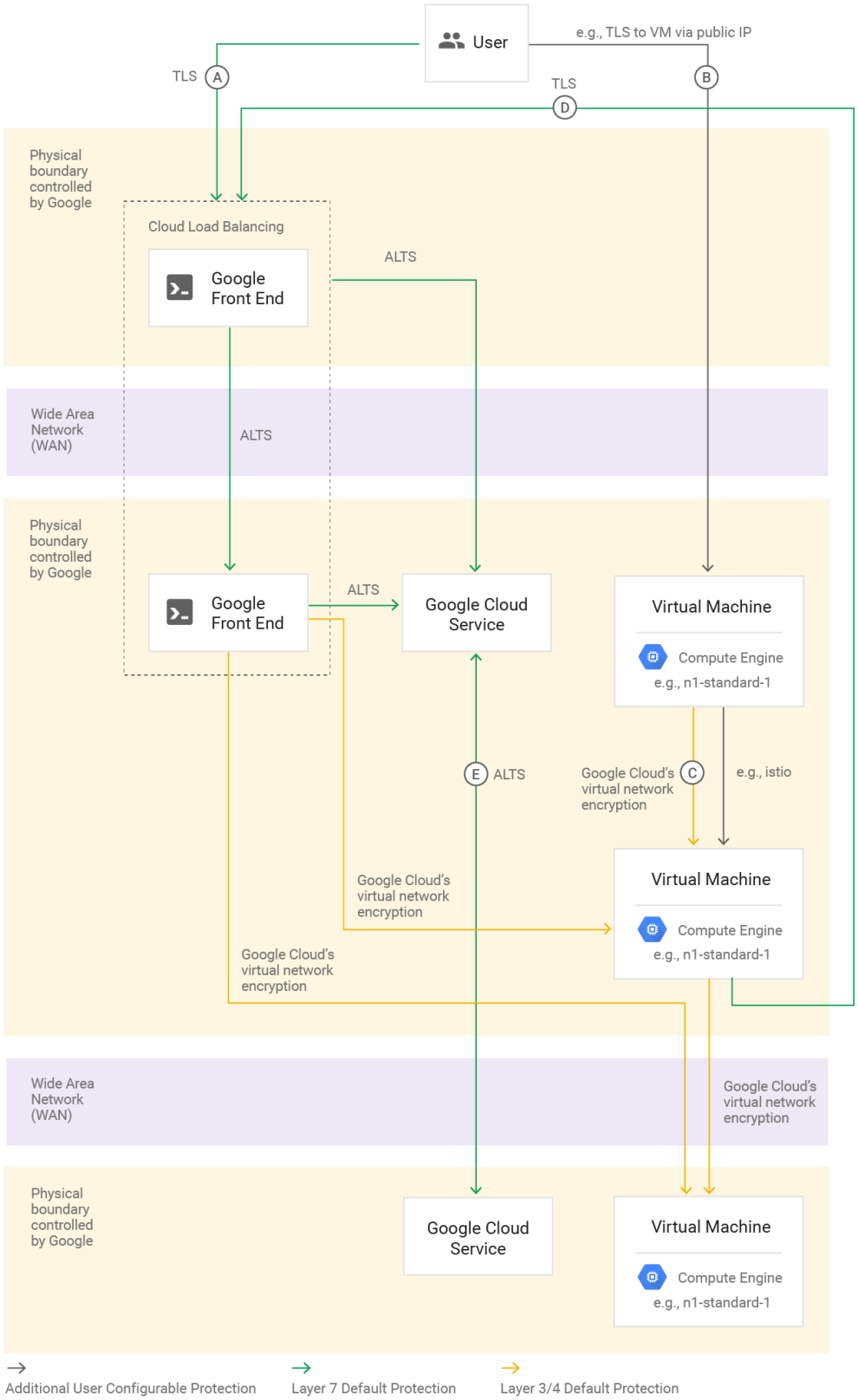

Figure 1: Protection by default and options overlaid on a VPC network

Encryption in Transit by Default

Google uses various methods of encryption, both default and user configurable, for data in transit. The type of encryption used depends on the OSI layer, the type of service, and the physical component of the infrastructure. Figures 2 and 3 below illustrate the optional and default protections Google Cloud has in place for layers 3, 4, and 7.

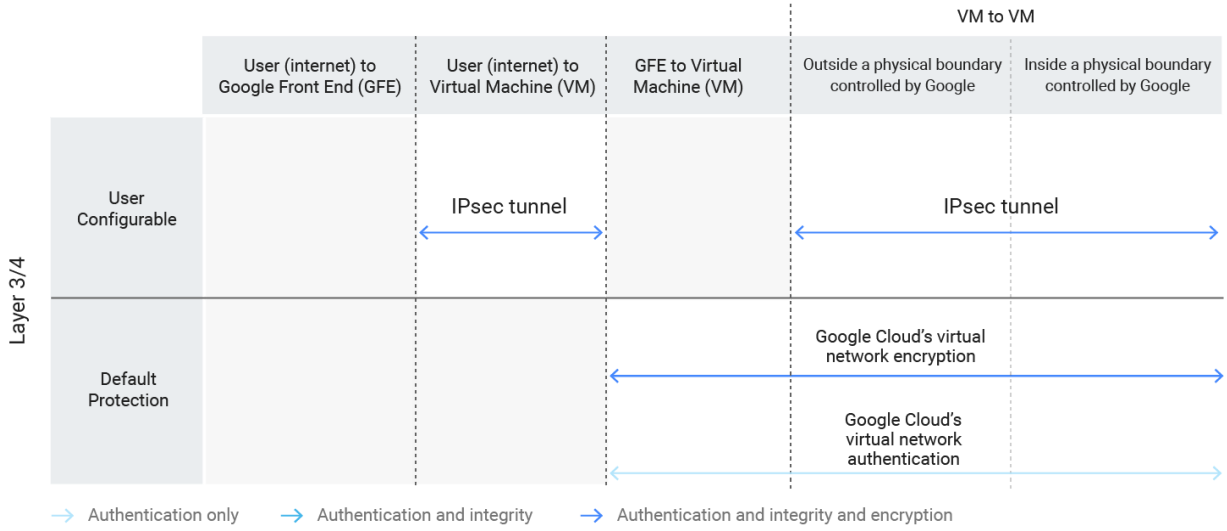

Figure 2: Protection by Default and Options at Layers 3 and 4 across Google Cloud

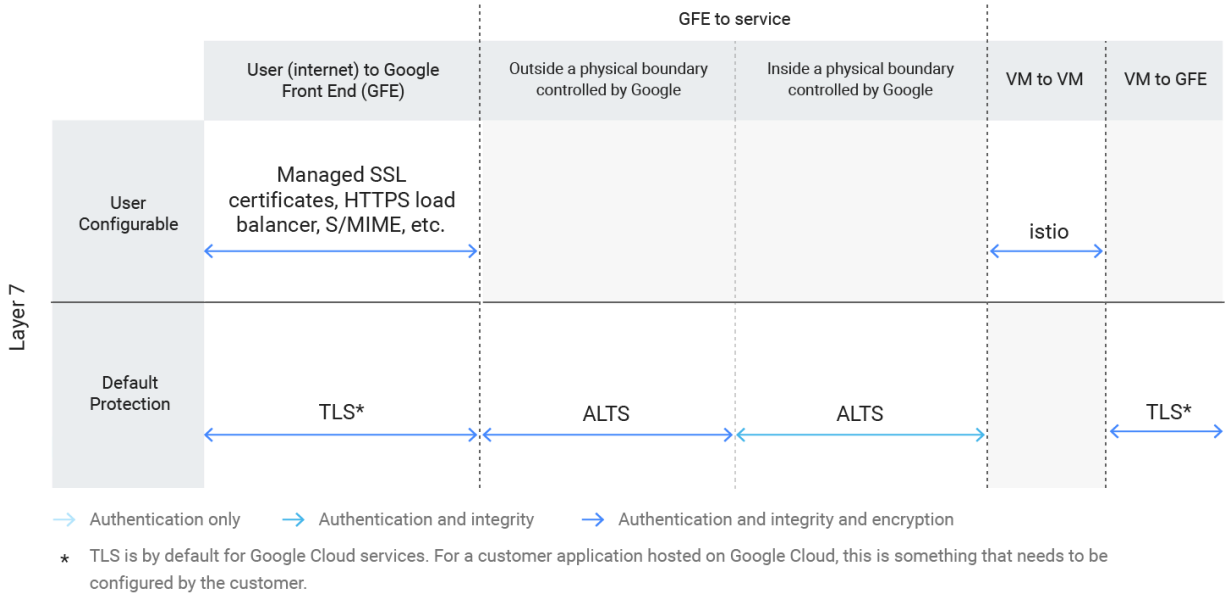

Figure 3: Protection by Default and Options at Layer 7 across Google Cloud[3]

The remainder of this section describes the default protections that Google uses to protect data in transit.

User to Google Front End encryption

Today, many systems use HTTPS to communicate over the Internet. HTTPS provides security by using a TLS connection, which ensures the authenticity, integrity, and privacy of requests and responses. To accept HTTPS requests, the receiver requires a public–private key pair and an X.509 certificate for server authentication from a Certificate Authority (CA). The key pair and certificate help protect a user's requests at the application layer (layer 7) by proving that the receiver owns the domain name for which requests are intended. The following subsections discuss the components of user to GFE encryption, namely: TLS, BoringSSL, and Google's Certificate Authority. Recall that not all customer paths route via the GFE; notably, the GFE is used for traffic from a user to a Google Cloud service, and from a user to a customer application hosted on Google Cloud that uses Google Cloud Load Balancing.

Transport Layer Security (TLS)

When a user sends a request to a Google Cloud service, we secure the data in transit; providing authentication, integrity, and encryption, using HTTPS with a certificate from a web (public) certificate authority. Any data the user sends to the GFE is encrypted in transit with Transport Layer Security (TLS) or QUIC. GFE negotiates a particular encryption protocol with the client depending on what the client is able to support. GFE negotiates more modern encryption protocols when possible.

GFE's scaled TLS encryption applies not only to end-user interactions with Google, it also facilitates API interactions with Google over TLS, including Google Cloud. Additionally, our TLS encryption is used in Gmail to exchange email with external mail servers (more detail in Require TLS in Gmail).

Google is an industry leader in both the adoption of TLS and the strengthening of its implementation. To this end, we have enabled, by default, many of the security features of TLS. For example, since 2011 we have been using forward secrecy in our TLS implementation. Forward secrecy makes sure the key that protects a connection is not persisted, so an attacker that intercepts and reads one message cannot read previous messages.

BoringSSL

BoringSSL is a Google-maintained, open-source implementation of the TLS protocol, forked from OpenSSL, that is mostly interface-compatible with OpenSSL. Google forked BoringSSL from OpenSSL to simplify OpenSSL, both for internal use and to better support the Chromium and Android Open Source Projects. BoringCrypto, the core of BoringSSL, has been validated to FIPS 140-2 level 1.

TLS in the GFE is implemented with BoringSSL. Table 1 shows the encryption protocols that GFE supports when communicating with clients.

Protocols	Authentication	Key exchange	Encryption	Hash Functions
TLS 1.3[4]	RSA 2048	Curve25519	AES-128-GCM	SHA384
TLS 1.2	ECDSA P-256	P-256 (NIST secp256r1)	AES-256-GCM	SHA256
TLS 1.1			AES-128-CBC	SHA1[8]
TLS 1.0[5]			AES-256-CBC	MD5[9]
QUIC[6]			ChaCha20-Poly 1305	
			3DES[7]	

Table 1: Encryption Implemented in the Google Front End for Google Cloud Services and Implemented in the BoringSSL Cryptographic Library

Google's Certificate Authority

As part of TLS, a server must prove its identity to the user when it receives a connection request. This identity verification is achieved in the TLS protocol by having the server present a certificate containing its claimed identity. The certificate contains both the server's DNS hostname and its public key. Once presented, the certificate is signed by an issuing Certificate Authority (CA) that is trusted by the user requesting the connection[10]. As a result, users who request connections to the server only need to trust the root CA. If the server wants to be accessed ubiquitously, the root CA needs to be known to the client devices worldwide. Today, most browsers, and other TLS client implementations, each have their own set of root CAs that are configured as trusted in their “root store”.

Historically, Google operated its own issuing CA, which we used to sign certificates for Google domains. We did not, however, operate our own root CA. Today, our CA certificates are cross-signed by multiple root CAs which are ubiquitously distributed, including Symantec (“GeoTrust”) and roots previously operated by GlobalSign (“GS Root R2” and “GS Root R4”).

In June 2017, we announced a transition to using Google-owned root CAs. Over time, we plan to operate a ubiquitously distributed root CA which will issue certificates for Google domains and for our customers.

Root key migration and key rotation

Root CA keys are not changed often, as migrating to a new root CA requires all browsers and devices to embed trust of that certificate, which takes a long time. As a result, even though Google now operates its own root CAs, we will continue to rely on multiple third-party root CAs for a transitional period to account for legacy devices while we migrate to our own.

Creating a new root CA requires a key ceremony. At Google, the ceremony mandates that a minimum 3 of the 6 possible authorized individuals physically gather to use hardware keys that are stored in a safe. These individuals meet in a dedicated room, shielded from electromagnetic interference, with an air-gapped Hardware Security Module (HSM), to generate a set of keys and certificates. The dedicated room is in a secure location in Google data centers. Additional controls, such as physical security measures, cameras, and other human observers, ensure that the process goes as planned. If the ceremony is successful the generated certificate is identical to a sample certificate, except for the issuer name, public key and signature. The resulting root CA certificate is then submitted to browser and device root programs for inclusion. This process is designed to ensure that the privacy and security of the associated private keys are well understood so the keys can be relied upon for a decade or more.

As described earlier, CAs use their private keys to sign certificates, and these certificates verify identities when initiating a TLS handshake as part of a user session. Server certificates are signed with intermediate CAs, the creation of which is similar to the creation of a root CA. The intermediate CA's certificates are distributed as part of the TLS session so it's easier to migrate to a new intermediate CA. This method of distribution also enables the CA operator to keep the root CA key material in an offline state.

The security of a TLS session is dependent on how well the server's key is protected. To further mitigate the risk of key compromise, Google's TLS certificate lifetimes are limited to approximately three months and the certificates are rotated approximately every two weeks.

A client that has previously connected to a server can use a private ticket key[11] to resume a prior session with an abbreviated TLS handshake, making these tickets very valuable to an attacker. Google rotates ticket keys at least once a day and expires the keys across all properties every 3 days. To learn more about session key ticket rotation, see Measuring the Security Harm of TLS Crypto Shortcuts.

Google Front End to Application Front Ends

In some cases, as discussed in How traffic gets routed, the user connects to a GFE inside of a different physical boundary than the desired service and the associated Application Front End. When this occurs, the user's request and any other layer 7 protocol, such as HTTP, is either

protected by TLS, or encapsulated in an RPC which is protected using Application Layer Transport Security (ALTS), discussed in Service-to-service authentication, integrity, and encryption. These RPCs are authenticated and encrypted.

For Google Cloud services, RPCs are protected using ALTS by default. For customer applications hosted on Google Cloud, if traffic is routed via the Google Front End, for example if they are using the Google Cloud Load Balancer, traffic to the VM is protected using Google Cloud's virtual network encryption, described in the next section.

Google Cloud's virtual network encryption and authentication

Encryption of private IP traffic within the same VPC or across peered VPC networks within Google Cloud's virtual network is performed at the network layer.

We use the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) with a 128-bit key (AES-128-GCM) to implement encryption at the network layer. Each pair of communicating hosts establishes a session key via a control channel protected by ALTS for authenticated and encrypted communications. The session key is used to encrypt all VM-to-VM communication between those hosts, and session keys are rotated periodically.

At the network layer (layer 3), Google Cloud's virtual network authenticates all traffic between VMs. This authentication, achieved via security tokens, protects a compromised host from spoofing packets on the network.

During authentication, security tokens are encapsulated in a tunnel header which contains authentication information about the sender and receiver. The control plane[12] on the sending side sets the token, and the receiving host validates the token. Security tokens are pre-generated for every flow, and consist of a token key (containing the sender's information) and the host secret. One secret exists for every source-receiver pair of physical boundaries controlled by or on behalf of Google.

Figure 4 shows how token keys, host secrets, and security tokens are created.

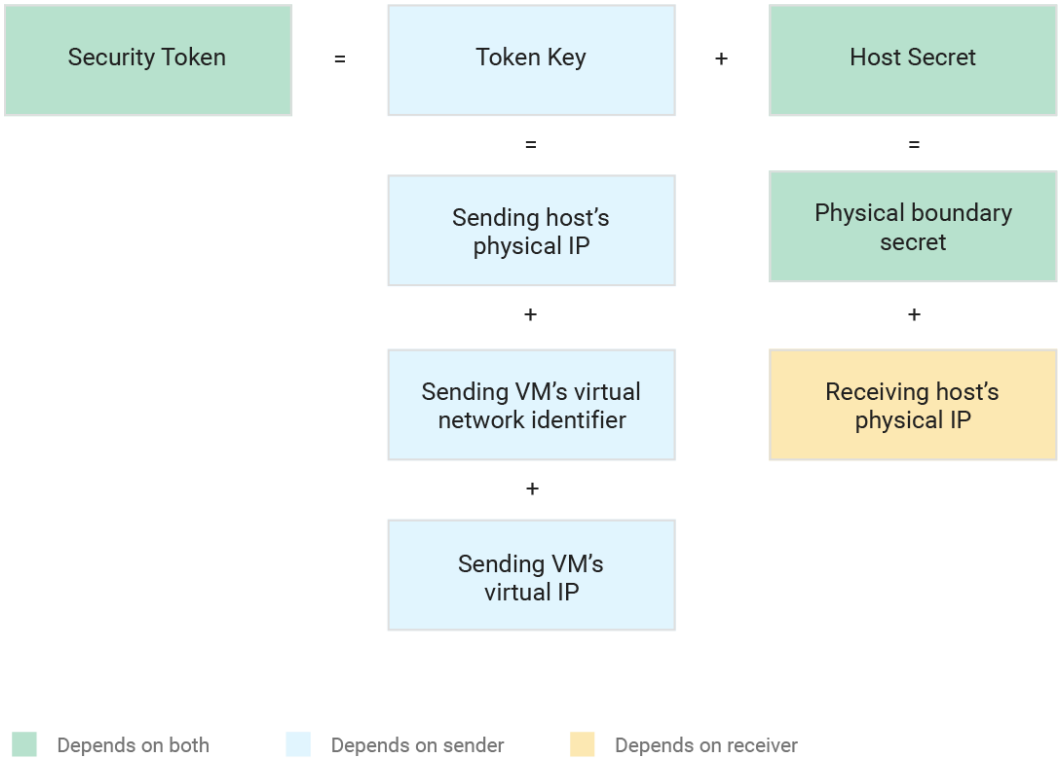

Figure 4: Security Tokens

The physical boundary secret is a 128-bit pseudorandom number, from which host secrets are derived by taking an HMAC-SHA1. The physical boundary secret is negotiated by a handshake between the network control planes of a pair of physical boundaries and renegotiated every few hours. The security tokens used for individual VM-to-VM authentication, derived from these and other inputs, are HMACs, negotiated for a given sender and receiver pair.

Service-to-service authentication, integrity, and encryption

Within Google's infrastructure, at the application layer (layer 7), we use our Application Layer Transport Security (ALTS) for the authentication, integrity, and encryption of Google RPC calls from the GFE to a service, and from service to service.

ALTS uses service accounts for authentication. Each service that runs in Google's infrastructure runs as a service account identity with associated cryptographic credentials. When making or receiving RPCs from other services, a service uses its credentials to authenticate. ALTS verifies these credentials using an internal certificate authority.

Within a physical boundary controlled by or on behalf of Google, ALTS provides both authentication and integrity for RPCs in “authentication and integrity” mode. For traffic over the WAN outside of physical boundaries controlled by or on behalf of Google, ALTS enforces

encryption for infrastructure RPC traffic automatically in “authentication, integrity, and privacy” mode. Currently, all traffic to Google services, including Google Cloud services, benefits from these same protections.

ALTS is also used to encapsulate other layer 7 protocols, such as HTTP, in infrastructure RPC mechanisms for traffic moving from the Google Front End to the Application Front End. This protection isolates the application layer and removes any dependency on the network path's security.

Services can be configured to accept and send ALTS communications only in “authentication, integrity and privacy” mode, even within physical boundaries controlled by or on behalf of Google. One example is Google's internal key management service, which stores and manages the encryption keys used to protect data stored at rest in Google's infrastructure.

ALTS Protocol

ALTS has a secure handshake protocol similar to mutual TLS. Two services wishing to communicate using ALTS employ this handshake protocol to authenticate and negotiate communication parameters before sending any sensitive information. The protocol is a two-step process:

- **Step 1: Handshake** The client initiates an elliptic curve-Diffie Hellman (ECDH) handshake with the server using Curve25519. The client and server each have certified ECDH public parameters as part of their certificate, which is used during a Diffie Hellman key exchange. The handshake results in a common traffic key that is available on the client and the server. The peer identities from the certificates are surfaced to the application layer to use in authorization decisions.
- **Step 2: Record encryption** Using the common traffic key from Step 1, data is transmitted from the client to the server securely. Encryption in ALTS is implemented using BoringSSL and other encryption libraries. Encryption is most commonly AES-128-GCM while integrity is provided by AES-GCM's GMAC.

The following diagram shows the ALTS handshake in detail. In newer implementations, a process helper does the handshake; there are still some cases where this is done directly by the applications.

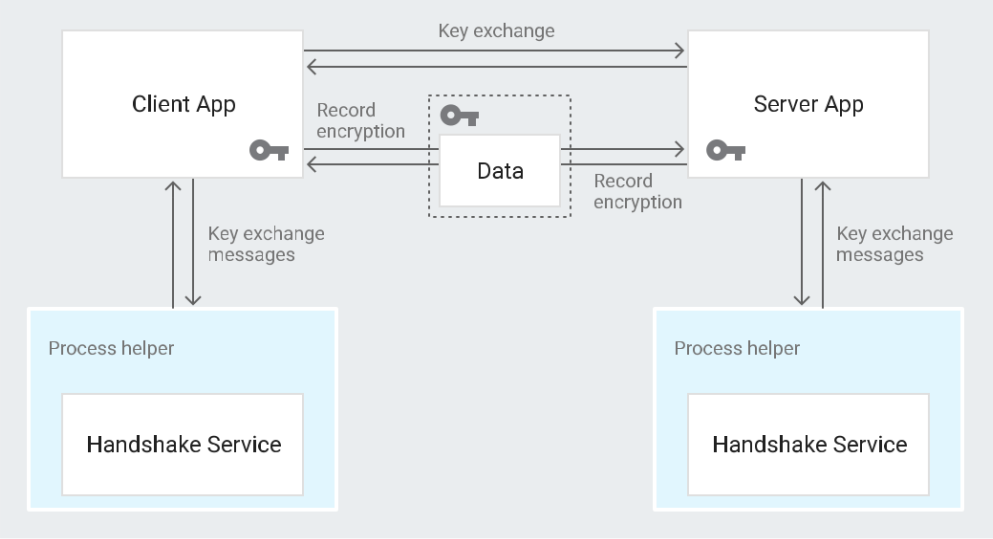

Figure 5: ALTS handshake

As described at the start of section Service-to-service authentication, integrity, and encryption, ALTS uses service accounts for authentication, with each service that runs on Google's infrastructure running as a service identity with associated cryptographic credentials. During the ALTS handshake, the process helper accesses the private keys and corresponding certificates that each client-server pair uses in their communications. The private key and corresponding certificate (signed protocol buffer) have been provisioned for the service account identity of the service.

ALTS Certificates

There are multiple kinds of ALTS certificate:

- **Machine certificates**: provide an identity to core services on a specific machine. These are rotated approximately every 6 hours.
- **User certificates**: provide an end user identity for a Google engineer developing code. These are rotated approximately every 20 hours.
- **Borg job certificates**: provide an identity to jobs running within Google's infrastructure. These are rotated approximately every 48 hours.

The root certification signing key is stored in Google's internal certificate authority (CA), which is unrelated and independent of our external CA.

Encryption in ALTS

Encryption in ALTS can be implemented using a variety of algorithms, depending on the machines that are used. For example, most services use AES-128-GCM[13]. More information on ALTS encryption can be found in Table 2.

Machines	Message encryption used	
Most common	AES-128-GCM	
Sandy Bridge or older	AES-128-VCM	Uses a VMAC instead of a GMAC and is slightly more efficient on these older machines.

Table 2: Encryption in ALTS

Most Google services use ALTS, or RPC encapsulation that uses ALTS. In cases where ALTS is not used, other protections are employed. For example:

- Some low-level machine management and bootstrapping services use SSH
- Some low-level infrastructure logging services TLS or Datagram TLS (DTLS)[14]
- Some services that use non-TCP transports use other cryptographic protocols or network level protections when inside physical boundaries controlled by or on behalf of Google

Communications between VMs and Google Cloud Platform services use TLS to communicate with the Google Front End, not ALTS. We describe these communications in Virtual machine to Google Front End encryption.

Virtual machine to Google Front End encryption

VM to GFE traffic uses external IPs to reach Google services, but you can configure Private Google Access feature to use Google-only IP addresses for the requests.

As with requests from an external user to Google, we support TLS traffic by default from a VM to the GFE. The connection happens in the same way as any other external connection. For more information on TLS, see Transport Layer Security (TLS).

User-configurable options for encryption in transit

Encryption in Transit described the default protections that Google has in place for data in transit. This section describes the configurations our users can make to these default protections.

On-premises data center to Google Cloud

TLS using GCLB external load balancers

If your cloud service uses a Google HTTPS or SSL Proxy external load balancer, then GFE terminates the TLS connections from your users using SSL certificates that you provision and control. More information on customizing your certificate can be found in our SSL Certificates documentation.

IPSec tunnel using Cloud VPN

As a Google Cloud customer, you can use Google Cloud VPN to securely connect your on-premises network to your Google Cloud VPC network through an IPSec VPN connection (layer 3). Traffic traveling between the two networks is encrypted by one VPN gateway and decrypted by the other VPN gateway. This protects your data over the Internet. In addition, you can set up multiple, load-balanced tunnels through multiple VPN gateways. The Google Cloud VPN protects your data in the following ways:

- Packets **from your VMs to the Cloud VPN gateway** remain within the VPC network. These packets are authenticated and encrypted by Google Cloud's virtual network.
- Packets **from the Cloud VPN to your on-premises VPN** are authenticated and encrypted using an IPSec tunnel.
- Packets f**rom your on-premises VPN to your on-premises hosts** are protected by whatever controls you have in place on your network.

To set up a VPN, create a Cloud VPN gateway and tunnel on the hosted service's VPC network, then permit traffic between the networks. You also have the option of setting up a VPN between two VPC networks.

You can further customize your network by specifying the Internet Key Exchange[15] (IKE) version for your VPN tunnel. There are two versions of IKE to choose from, IKEv1 and IKEv2, each of which supports different ciphers. If you specify IKEv1, Google encrypts the packets using AES-128-CBC and provides integrity through SHA-1 HMAC[16]. For IKEv2, a variety of ciphers are available and supported. In all cases, Google Cloud VPN will negotiate the most secure

common protocol the peer devices support. Full instructions on setting up a VPN can be found in our documentation Choosing a VPN Routing Option.

An alternative to a Cloud VPN (IPSec) tunnel is Google Cloud Dedicated Interconnect. Dedicated Interconnect provides direct physical connections and private IP communication between your on-premises network and your VPC network. The data traveling over Dedicated or Partner interconnect is NOT encrypted by default and should be secured at the application layer, using TLS for example. MACsec (layer 2 protection) is not currently supported.

User to Google Front End

Managed SSL certificates: Free and automated certificates

When building an application on Google Cloud, you can leverage GFE's support of TLS by configuring the SSL certificate you use. For example, you can have the TLS session terminate in your application. This termination is different to the TLS termination described in TLS using GCLB external load balancers.

Google also provides free and automated SSL certificates in both the Firebase Hosting and Google App Engine custom domains. These certificates are only available for Google-hosted properties. With Google App Engine custom domains, you can also provide your own SSL certificates and use an HTTP Strict Transport Security (HSTS) header.

Once your domain is pointed at Google's infrastructure, we request and obtain a certificate for that domain to allow secure communications. We manage the TLS server private keys, which are either 2048-bit RSA or secp256r1 ECC, and renew certificates on behalf of our customers.

Require TLS in Gmail

As discussed in Transport Layer Security, Gmail uses TLS by default. Gmail records and displays whether the last hop an email made was over a TLS session[17]. When a Gmail user exchanges an email with another Gmail user, the emails are protected by TLS, or in some cases, sent directly within the application. In these cases, the RPCs used by the Gmail application are protected with ALTS as described in Service-to-service authentication, integrity, and encryption. For incoming messages from other email providers, Gmail does not enforce TLS. Gmail administrators can configure Gmail to require a secure TLS connection for all incoming and outgoing emails.

Gmail S/MIME

Secure/Multipurpose Internet Mail Extensions (S/MIME) is an email security standard that provides authentication, integrity, and encryption. The implementation of the S/MIME standard mandates that certificates associated with users sending emails are hosted in a public CA.

As an administrator, you can configure Gmail to enable S/MIME for outgoing emails, set up policies for content and attachment compliance, and create routing rules for incoming and outgoing emails. Once configured, you must upload users' public certificates to Gmail using the Gmail API. For users external to Gmail, an initial S/MIME-signed message must be exchanged to set S/MIME as the default.

Service-to-service and VM-to-VM encryption

Istio is an open-source service mesh developed by Google, IBM, Lyft, and others, to simplify service discovery and connectivity. Istio authentication provides automatic encryption of data in transit between services, and management of associated keys and certificates. Istio can be used in Google Kubernetes Engine and Google Compute Engine.

If you want to implement mutual authentication and encryption for workloads, you can use istio auth. Specifically, for a workload in Kubernetes, Istio auth allows a cluster-level CA to generate and distribute certificates, which are then used for pod-to-pod mutual Transport Layer Security (mTLS).

How Google helps the Internet encrypt data in transit

Encryption in Transit by Default and User-configurable options for encryption in transit explained the default and customizable protections Google Cloud has in place for customer data in transit. In addition, Google has several open-source projects and other efforts that encourage the use of encryption in transit and data security on the Internet at large.

Certificate Transparency

As discussed in User to Google Front End encryption, to offer HTTPS, a site must apply first for a certificate from a trusted web (public) Certificate Authority (CA). The Certificate Authority is responsible for verifying that the applicant is authorized by the domain holder, as well as ensuring that any other information included in the certificate is accurate.This certificate is then presented to the browser to authenticate the site the user is trying to access. In order to ensure HTTPS is properly authenticated, it's important to ensure that CAs only issue certificates that the domain holder has authorized.

Certificate Transparency (CT) is an effort that Google launched in March 2013 to provide a way for site operators and domain holders to detect if a CA has issued any unauthorized or incorrect certificates. It works by providing a mechanism for domain holders, CAs, and the public to log the trusted certificates they see or, in the case of CAs, the certificates they issue, to publicly verifiable, append-only, tamper-proof logs. The certificates in these logs can be examined by anyone to ensure the information is correct, accurate, and authorized.

The first version of Certificate Transparency was specified in an IETF experimental RFC, RFC 6962. During the development of Certificate Transparency, Google open-sourced a number of tools, including an open-source log server that can record certificates, as well as tools to create Certificate Transparency logs. In addition, Google Chrome requires that some certificates must be publicly disclosed, such as for Extended Validation (EV) certificates or certificates issued from CAs that have improperly issued certificates in the past. From 2018, Chrome will require that all new publicly trusted certificates be disclosed.

As a site operator, you can use Certificate Transparency to detect if unauthorized certificates have been issued for your website. A number of free tools exist to make this easy to do, such as Google's Certificate Transparency Report, Certificate Search, or tools from Facebook. Even if you don't use Certificate Transparency, a number of browsers now examine Certificate Transparency regularly to ensure that the CAs your users trust to access your website are adhering to industry requirements and best practices, reducing the risk of fraudulent certificates being issued.

Increasing the use of HTTPS

As described in User to Google Front End encryption, we work hard to make sure that our sites and services provide modern HTTPS by default. Our goal is to achieve 100% encryption across our products and services. To this end, we publish an annual HTTPS Transparency Report that tracks our progress towards our goal for all properties, including Google Cloud. We continue to work through the technical barriers that make it difficult to support encryption in some of our products, such as solutions for browsers or other clients that do not support HTTP Strict Transport Security (HSTS)[18]. We use HSTS for some of our sites, including the google.com homepage, to allow users to connect to a server only over HTTPS.

We know that the rest of the Internet is working on moving to HTTPS. We try to facilitate this move in the following ways:

- We provide developers with advice on why HTTPS matters, how to enable HTTPS, and best practices when implementing HTTPS
- We have created tools in Chrome like the Security panel in DevTools to help developers assess the HTTPS status of their site(s)

- We financially support the Let's Encrypt initiative that allows anyone to obtain a free certificate for their website. Google representatives sit on the technical advisory board of Let's Encrypt's parent organization, Internet Security Research Group

In 2016, we began publishing metrics on “HTTPS usage on the Internet” for the Top 100 non-Google sites on the Internet. With these metrics, we aim to increase awareness and help make the Internet a safer place for all users. In October 2017, Chrome formally renewed its financial support of Let's Encrypt as a Platinum sponsor.

Increasing the use of secure SMTP: Gmail indicators

Most email is exchanged using the Simple Mail Transfer Protocol (SMTP) which, by default, sends email without using encryption. To encrypt an email, the mail provider must implement security controls like TLS.

As discussed in User to Google Front End encryption, Gmail uses TLS by default. In addition, Require TLS in Gmail describes how Gmail administrators can enforce the use of TLS protection for incoming and outgoing emails. Like Google's efforts with HTTPS transparency, Gmail provides data on TLS use for incoming emails to Gmail. This data is presented in our Safer Email Transparency Report.

Google, in partnership with the IETF and other industry key players, is leading the development of SMTP STS. SMTP STS is like HSTS for HTTPS, forcing the use of SMTP over only encrypted channels.

Chrome APIs

In February 2015, Chrome announced that powerful new features will be available only to secure origins[19]. Such features include the handling of private information and access to sensors on a user's device. Starting with geolocation in Chrome 50, we began deprecating these features for insecure origins.

Ongoing Innovation in Encryption in Transit

Chrome Security User Experience

Google Chrome is an industry leader in leveraging its UI to display security information in ways that allow users to quickly understand the safety of their connection to a site. With this information, users can make informed decisions about when and how they share their data. Chrome conducts extensive user research, the results of which are shared in peer-reviewed papers.

To help further protect its users, Chrome has announced that by the end of 2017, it will mark all HTTP connections as non-secure. Starting with Chrome 56, by default, users will see a warning if an HTTP page includes a form with password or credit card fields. With Chrome 62, a warning will be shown when a user enters in data on an HTTP page, and for all HTTP pages visited in Incognito mode. Eventually, Chrome will show a warning for all pages that are served over HTTP.

To see how particular configurations are displayed to users in Chrome, you can use the BadSSL tool.

Key Transparency

A significant deterrent to the widespread adoption of message encryption is the difficulty of public key exchange: how can I reliably find the public key for a new user with which I am communicating? To help solve this issue, in January 2017, Google announced Key Transparency. This is an open framework that provides a generic, secure, and auditable means to distribute public keys. The framework removes the need for users to perform manual key verification. Key Transparency is primarily targeted at the distribution of users' public keys in communications, for example, E2E and OpenPGP email encryption. Key Transparency's design is a new approach to key recovery and distribution and is based on insights gained from Certificate Transparency and CONIKS.

Key Transparency's development is open-source and it is implemented using a large-scale Merkle tree. Key Transparency Verification allows account owners to see what keys have been associated with their accounts and how long an account has been active and stable. The long-term goal of Google's Key Transparency work is to enable anyone to run a Key Transparency server and make it easy to integrate into any number of applications.

Post-quantum cryptography

Google plans to remain the industry leader in encryption in transit. To this end, we have started work in the area of post-quantum cryptography. This type of cryptography allows us to replace existing crypto primitives, that are vulnerable to efficient quantum attacks, with post-quantum candidates that are believed to be more robust. In July 2016 we announced that we had conducted an experiment on the feasibility of deploying such an algorithm by using the New Hope post-quantum crypto algorithm in the developer version of Chrome. In addition to this work, researchers at Google have published papers on other practical post-quantum key-exchange protocols.

Appendix

Read more about Google Cloud Security, including our Infrastructure Security Design Overview; as well as Google Cloud compliance, including the public SOC 3 audit report.

Footnotes

[1] Partner solutions include both solutions offered in Cloud Launcher, as well as products built in collaboration with partners, such as Cloud Dataprep.
[2] You can still disable this encryption, for example for HTTP access to Google Cloud Storage buckets.
[3] VM-to-Service communications not protected at Layer 7 are still protected at layers 3 and 4
[4] TLS 1.3 is not yet finalized. The draft version is implemented only for certain Google domains for testing, such as Gmail.
[5] Google supports TLS 1.0 for browsers that still use this version of the protocol. Note that any Google site processing credit card information will no longer support TLS 1.0 by July 2018 when Payment Card Industry (PCI) compliance requires its deprecation.
[6] For details on QUIC, see https://www.chromium.org/quic.
[7, 8, 9] For backwards compatibility with some legacy operating systems, we support 3DES, SHA1 and MD5
[10] In the case of chained certificates, the CA is transitively trusted.
[11] This could be either a session ticket RFC 5077 or a session ID RFC 5246.
[12] The control plane is the part of the network that carries signalling traffic and is responsible for routing.
[13] Previously, other protocols were used but are now deprecated. Less than 1% of jobs use these older protocols.
[14] Datagram TLS (DTLS) provides security for datagram-based applications by allowing them to communicate in a way that prevents eavesdropping and tampering.
[15] Internet Key Exchange (IKE) is the protocol used to set up a security association in the IPSec protocol suite.
[16] HMAC-SHA-1 is not broken by a SHA-1 collision, such as the SHAttered collision Google researchers found.
[17] For Enterprise Plus, this isn't shown in the UI. Domain administrators can examine data for their domain using Email Log Search.
[18] HTTP Strict Transport Security (HSTS) is a mechanism enabling web sites to declare themselves accessible only via secure connections and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections.
[19] Secure origins are connections that match certain scheme, host, or port patterns.