Android Cloud Backup/Restore

Google

October 10, 2018 – Version 1.0

Prepared for

Troy Kensinger
Shabsi Walfish

Prepared by

Mason Hemmel
Jason Meltzer
Thomas Pornin
Keegan Ryan
Javed Samuel
David Wong
Rob Wood
Greg Worona

Public Summary

Overview

In the summer of 2018, Google engaged NCC Group to conduct a security assessment of the Android Cloud Backup/Restore feature, which premiered in Android Pie. This engagement focused on a threat model that included attacks by rogue Google employees (or other malicious insiders) with privileges up to and including root-in-production. The Android backup/restore feature is only one potential use-case for the key protection mechanisms which are part of Google's Cloud Key Vault project, so the overall goal of our work was to analyze the security of the component systems and services of Google Cloud Key Vault as a whole. While NCC Group analyzed Google Cloud Key Vault's security posture holistically, the team most closely examined threats which could result in gaining access to cryptographic key material used to protect the user's backup data.

NCC Group conducted this assessment between July 9 and August 3, 2018, primarily on-site at Google's Mountain View, CA campus.

Scope

- Assessment of the design and implementation of cryptographic mechanisms and protocols in:
  - Google Cloud Key Vault Android Client
  - Google Cloud Key Vault Service
  - Google Cloud Key Vault/Titan Hardware and Firmware
  - Google Cloud Key Vault service and systems deployment
- Assessment of overall design and implementation of Google Cloud Key Vault/Titan Hardware and Firmware
- Audit of system provisioning and site deployment

Positive Design Elements

Throughout the assessment, NCC Group was impressed by both the well-rounded design and the high-quality code which took security into consideration. Numerous possible avenues of achieving a compromise were investigated and most of these ended with a determination that the design and implementation were already taking the particular attack into account and had sufficient mitigations. Some notable highlights include:

- **Side-channel resistance in the cryptographic accelerators** was observed. This is a relatively uncommon practice outside the smartcard space, and shows an attention to detail in the face of a physical adversary. Both power and timing side channels appear to have been considered (for instance, the implementation of elliptic curve operations avoids all data-dependent conditional jumps, and applies random masking repeatedly to thwart power analysis).
- **Cryptographic review** of some parts of the cryptographic implementations was performed, notably formal validation of some operations on finite fields, as used in the implementation of elliptic curve operations. It was observed that these cryptographic implementations were done in a secure manner.
- **A fuzzer was implemented** by Google staff for the Google Cloud Key Vault command interface. This is the primary external attack surface of the firmware, and even though the code was rigorously reviewed, it is possible for fuzzing to catch interesting errors which could have a security impact.
- **Tamper-resistance features** were included in the Titan hardware design. This includes protections against micro-probing, side-channel attacks, and fault injection through a variety of techniques. While evaluating these defenses was beyond the scope of this assessment, the presence of these features in the hardware is a great first step.
- **Explicit control flow** guards were found throughout the BootROM code. This is a strong mitigation against many fault injection attacks.
- **Strong auditing** mechanisms were built into all aspects of the system. Both the secure firmware update log feature and the always-active *prober* tool used for system health monitoring are examples of this design mindset. The ability to monitor the system allows any compromise of the system to be detected immediately and therefore suitable response procedures can be enacted. This functionality was relied upon during the Google Cloud Key Vault deployment audit as discussed below.

Limitations

- **Supply chain attacks** are not considered. For instance, NCC Group did not consider the threat arising from a Titan device loaded with compromised software prior to being provisioned in the HSM provisioning room. We note that such an attack could theoretically compromise the first Google Cloud Key Vault firmware load, which is crucial to the security guarantees of the device operation and all subsequent firmware loads. Nevertheless, this is a known and accepted weakness of the Google Cloud Key Vault V1 deployment.
- **Hardware attacks** directly against the Titan chip were also out of scope for this assessment. While NCC Group examined the general security properties of the chip during audits of the BootROM, BootLoader, and Google Cloud Key Vault firmware, the team did not assess the full details of the chip fuse configurations or the effectiveness of the anti-tamper countermeasures. Although teams both internal and external to Google have performed tests evaluating the risk posed by non-invasive side-channel and fault injection attacks, those results were not evaluated in this assessment.
- **The Google Cloud Key Vault device deployment** was audited only partially. Specifically, NCC Group visited a single data center and validated two of the cards among the cohorts there. While there is no reason to suspect that the other cohorts were deployed any differently, the Group makes note of this due to the fact that a firmware limitation at the time of the assessment meant that we could only attest to the correct provisioning of a given cohort after verifying each card in the cohort. Newer versions of the firmware allow for validation of entire cohort provisioning from a single member as long as one trusts each device's attestation keys.
- **All assessment was done in cooperation with Google**. NCC Group assumes goodwill on the part of Google staff, in particular that they were not trying to actively subvert our investigations, especially regarding the on-site visit to the data center. Throughout the engagement Google staff was co-operative and forthcoming with all information that was requested.

Key Findings

Two critical issues were found in the Google Cloud Key Vault firmware. One allowed arbitrary memory reads from the chip's internal memory which could have compromised the secret key material and thus allowed for user data recovery. The second one permitted an attacker to bypass the brute-force mitigations and attempt unlimited LSKF guesses. Both of the firmware issues were quickly fixed by Google engineers during the assessment and the fixes were reviewed to NCC Group's satisfaction.

Beyond these issues, several low severity issues were reported that do not pose an immediate threat to the system or to the users (all findings have been reviewed and are being prioritized accordingly). As well, several informational suggestions were made for future releases that would improve the defense-in-depth, correctness and security posture of the system.

Google Cloud Key Vault Deployment Audit

On August 1, 2018, NCC Group visited a Google data center in order to inspect an example deployment of the Google Cloud Key Vault Gneiss hardware. During the visit NCC Group selected two Google Cloud Key Vault servers with Gneiss cards from two separate machine clusters. The first Gneiss card was expected to be running the second-to-latest firmware (version 9983) and in the process of draining clients participating in the Android P beta (`ilby1`), while the second was expected to be running the latest Google Cloud Key Vault firmware (version 9980) and not yet actively serving clients (`ilez1`).

Our tests confirmed that the firmware loaded on the cards matched the versions that were expected to be running, which were binaries NCC Group was able to reproduce from the source code inspected during the audit. Google Cloud Key Vault firmware is built within the ChromiumOS build environment to enable reproducible builds.

Physical Security

The Google data center is dedicated facility that houses Google infrastructure exclusively. The physical security mechanisms in place at the data center were multi-layered, extensively monitored, and otherwise consistent with what Google

publicly describes in their Cloud service documentation.[1] After reviewing all the training materials and policy required for working in the data-center, NCC Group did not observe any instances while on-site where controls were intentionally bypassed or where the behavior of operations personnel didn't rigorously comply with documented processes.

Secure Destruction

Google has on-site facilities and accompanying processes to securely destroy media at their data centers and we were able to confirm that these facilities existed as we describe in the "Secure Scrap Process" sub-section of System Security Architecture on page 7. We also confirmed that they are well aware of the process for securely decommissioning Gneiss cards during our discussions with a local HWOps technician.

Testing Results

Testing results illustrate that our sample Google Cloud Key Vault hardware was running the firmware we expected. The test operates by dumping the firmware images (both A and B partitions) from the Titan chip's internal flash memory and comparing it against known-good copies.

Public Keys

Tests were also run to extract the public cohort keys from the device. These are expected to match those that are publicly published by Google.

```
-=-=-=-=-=-=-=-=-=-=-=-
Get cohorts public keys
-=-=-=-=-=-=-=-=-=-=-=-
Get cohort size: 3
Cohort 0 public key:
0000000 c104 b379 676e 780b 7f8f 9a3f 384f b388
0000010 d253 fee3 7fcc 14ff 6212 108a 6cf4 b723
0000020 1902 3514 64cc a589 582a 4de0 a8fd 394f
0000030 040b 1bd4 cd00 7b87 8058 9ecb c8c8 2a11
0000040 00d3
0000041

Cohort 1 public key:
0000000 bb04 15de 1e45 8d9d 390a a2c9 e589 73c2
0000010 c154 cbf0 9d51 2e5d aced 7e0a 1571 f457
0000020 065a 195e d29a b54c 99d3 454e e5ef 0557
0000030 aa88 f223 f02b 15d3 b42a 9343 ef11 d582
0000040 007f
0000041

Cohort 2 public key:
0000000 1604 d2d6 5116 a252 dbb4 bc1d 135c b1a2
0000010 17ba b38b e20d c8b0 5b60 7410 a49b 27ec
0000020 51c9 f965 6ac9 5c8d 300f 6940 1d1f 9a44
0000030 8426 7300 4c2d c400 02b8 663b 041b 0c82
0000040 00c7
0000041

Cohort 3 public key:
```

[1] Google Infrastructure Security Design Overview

Threat Model

Google Cloud Key Vault is a complex system with several moving parts that conceptually reduces to an end-to-end secure protocol between a user's phone and the Titan. While this reduction greatly reduces the attack surface, it is good to remind ourselves what the overall threat model looks like. While the current application of the Google Cloud Key Vault service only covers the user's backup encryption keys, there are ongoing discussions about extending this service to other use cases in the future, and thus the impacts are expected to increase.

External attackers: These are agents outside of Google which are directly targeting the technical infrastructure of the Google Cloud Key Vault service, its internal mechanisms or the client implementation run by Android phones. The cryptography tying a client to a Titan chip can be attacked at either end of the protocol. Addressing these threats on the Google Cloud Key Vault side involves a range of controls, from all the usual mechanisms that Google employs for applying security policy throughout their infrastructure and services (e.g. network security access controls, application tokens, and challenge-response mechanisms), to the specific mechanisms in the Google Cloud Key Vault service, such as the guess counter and hardware-based key management. Addressing these threats on the client side means having domain-separated protocols that disallow interaction between different applications, safe APIs, correct use of cryptography, and robust handling of secrets.

Internal attackers: The primary attacker in the threat model for Google Cloud Key Vault is a rogue Google employee, or another malicious insider, who, without the user's authorization, wants to decrypt a specific user's recovery keys (and eventually their backup data). The privilege level of the potential attacker includes administrators with "Root in Production" (Root-in-Prod) access to Google Cloud Key Vault systems as well as anyone with physical access to the Google Cloud Key Vault dedicated systems.

Note that an active Google attacker has many ways of compromising an active user, and the Google Cloud Key Vault protocol **does not protect** users from the following internal attacks:

Malicious Android software updates. The threat of Google insiders deploying malicious software on a device and extracting its keystore locally on the user's endpoint is considered out of scope. Google is already in possession of the Android code signing keys, and controls the firmware update process itself. Malicious updates will always be a plausible threat and preventing them is not an objective for the Google Cloud Key Vault design.

Denial-of-Service. Google will be responsible for storing all the data associated with the Android Cloud Encrypted Backup feature, which includes the encrypted backup data as well as the Google Cloud Key Vault data. This storage is done at Google's sole discretion. Preventing Google from intentionally deleting encrypted data is not possible. Other attacks, or administrative actions, that might result in a denial of recovery and which could be perpetrated by Google insiders are, likewise, considered out of scope. This category includes everything from blacklisting specific users to discontinuing entire services.

Design Assumptions

NCC Group assessed Google Cloud Key Vault version 1, which makes the following **assumptions** in its design:

- A user's backup data is protected by recovery keys bound to a Lock Screen Knowledge Factor (LSKF) that is backed up on Google servers and protected by an honest (well-provisioned) Google Cloud Key Vault cohort before an attack begins.
- An attacker could obtain a Titan from the relevant cohort by stealing one of the hosting Gneiss cards and then, perform a physically invasive attack on the chip in order to attempt recovery of secrets.
- An attacker could obtain any information that is stored on Google servers (including user backup and keys wrapped by the Google Cloud Key Vault protocol).
- An attacker can not obtain the user LSKF or trigger custom Android updates on the user's device.
- The supply chain for Gneiss cards used in the Google Cloud Key Vault service is unlikely to have been compromised prior to provisioning the cards.

The **security objectives** for Google Cloud Key Vault version 1, include the following:

- A user's LSKF must remain confidential at all times. It has low entropy, which calls for special care when using the

LSKF as input to cryptographic processing (i.e. as an encryption key) and handling the output of such operations.
- User application data must remain confidential: the backup data should be recoverable only by its owner, based on their possession of the LSKF.
- User application data integrity must be preserved: when a user restores from a backup, that data should be exactly what the user's device uploaded in the most recent backup and any alterations to that data should be reliably detected.
- The system should preserve availability: an external attacker should not be able to temporarily or permanently prevent recovery for individual or multiple users.

Google Cloud Key Vault Service Life-Cycle

The overall Google Cloud Key Vault Service itself relies on a number of assets:

- The list of Google Cloud Key Vault cohort public keys prior to signing.
- The Root of Trust used for signing the list of Google Cloud Key Vault public keys.
- Google Cloud Key Vault cohort shared secrets.
- Google Cloud Key Vault Titan chips.

Attackers may try to impact the process at various stages, notably:

- Alterations to Titan firmware or server source code prior to build and build artifacts prior to deployment
- Physical alterations of the Gneiss cards before and after provisioning with Google Cloud Key Vault firmware and the cohort provisioning ceremony
- Modifications to the Titan firmware at any point after initial flashing
- Passive and active analysis of the Titan chip processing from a close vantage point (Google Cloud Key Vault server under hostile control, trying to recover secret data held on the chip)
- Hostile control of the Google Cloud Key Vault servers and/or backup servers at any stage of the process (including during user enrollment, backup, and recovery)
- Network-based attacks, targeting either or both of the Google servers and the user devices
- Interference from malicious applications on the user device, at any stage of the process

The Google Cloud Key Vault protocol flows through several parts of the Google architecture and can be conceptually reduced to an end-to-end encrypted session between an Android phone and the Titan chip. This protocol flow is depicted in the diagram below. Thus, the security of the protocol can be modeled by analyzing both endpoints as well as an all-powerful man-in-the-middle adversary.

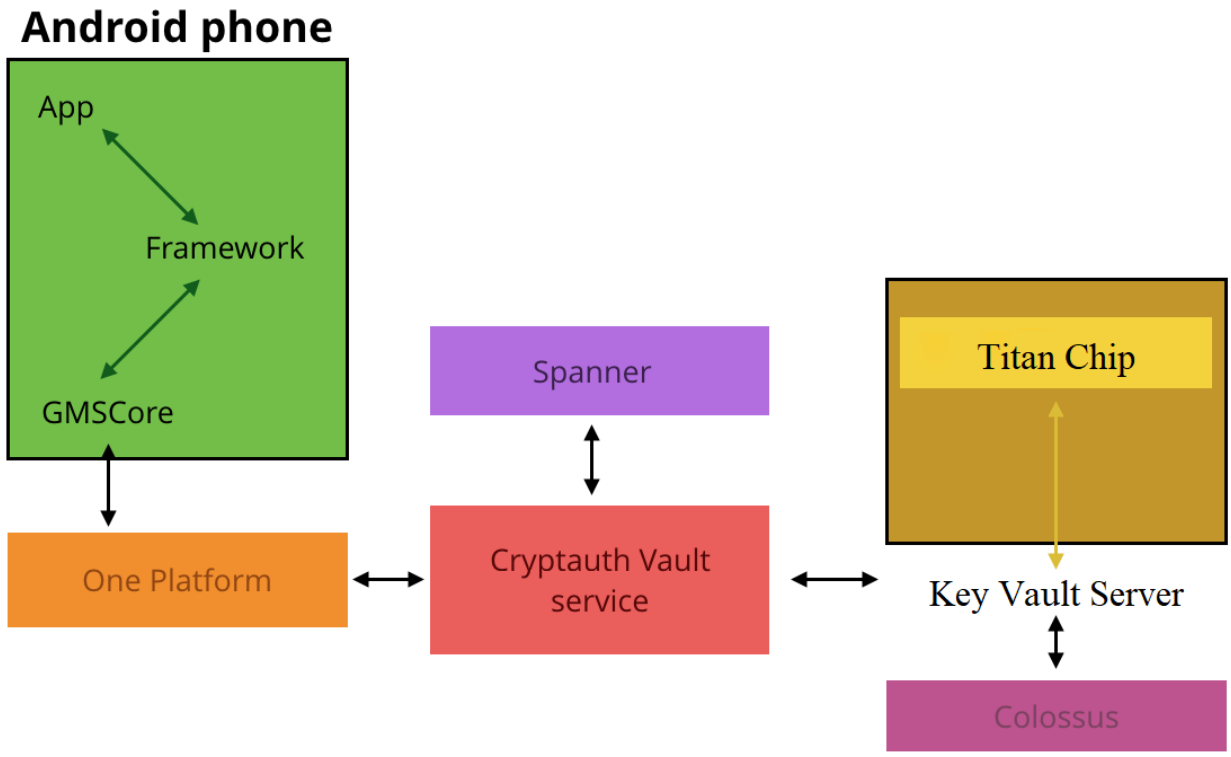


Figure 1: The Google Cloud Key Vault protocol architecture

Google Cloud Key Vault service

The Google Cloud Key Vault service uses secure hardware to protect a user's "Google Keychain". Only the secure hardware ever sees the user's secrets and the user must prove that they know this secret before a restore of the Keychain to a new device is completed. Some crucial actions performed by the Google Cloud Key Vault service include verifying the claims, determining the cohort public keys that are hosted, obtaining the Titan chip's public attestation key, decrypting a challenge encrypted to the Titan chip's public attestation key, and returning the current value of the bad guess counter.

Gneiss Card

The secure hardware deployed in the Google data center is centered around the Gneiss card which is a PCIe plug-in card. The image below shows a sample Gneiss card. Note that in the Google Cloud Key Vault threat model, none of the components on the PCB can be trusted except the Titan chip itself.

Figure 2: Gneiss card

Titan Secure Microcontroller

Titan is a security-focused microcontroller designed by Google that is built around an ARM SecureCore SC300 (Cortex-M3 based). It includes a hardware cryptographic accelerator and both anti-tamper and anti-side-channel features as well as secure boot and authenticated firmware update functionality. Titan was designed with special capabilities for SPI filtering and passthrough. One of its primary application areas is for protections of server platforms and device firmware that resides in SPI EEPROMs.

This microcontroller is the secure endpoint within the Google Cloud Key Vault service that handles all user authentication when decrypting recovery keys. The Titan hardware, and its Google Cloud Key Vault specific firmware, protect a number of local cryptographic assets described below:

- **Hardware key ladder**. This is a hardware-based key derivation system that allows both hardware and software derived values to be mixed into the key derivation. This produces device-unique keys that are inaccessible to the firmware and provides a root of trust for the Titan device.
- **Google Cloud Key Vault attestation private key**. This key is generated within the Titan chip on first boot. Its public key is exported to all other members of the cohort during the provisioning process. Sensitive data (such as the provisioning seeds and challenges) is then encrypted with the public key of other cohort members before being transmitted. An attacker may be able to MitM this data exchange; however such attacks would be caught later by the Google Cloud Key Vault Service's prober tool. If these keys are compromised by some other means (such as a passive side-channel attack), then the encrypted data could be compromised and the cohort keys recovered.
- **Google Cloud Key Vault cohort member shared symmetric secrets**. This key is used to protect the Merkle tree that

stores all of the per-user counters. A compromise of these keys would allow an attacker to have unlimited decryption attempts.

- **Google Cloud Key Vault cohort shared asymmetric secret** (derived from shared symmetric secrets). The public keys are published and used by GmsCore to encrypt the user's backup recovery keys. The private keys are held within the Google Cloud Key Vault device, stored in flash in a format that is wrapped with a Titan device-unique key. A compromise of this private key would allow user data to be decrypted.
- **Titan firmware upgrade keys**. This key is generated anew on each Google Cloud Key Vault firmware update. It is used to attest the firmware update logs. A compromise of this key would allow rogue firmware to be installed on a Google Cloud Key Vault device which could then compromise the integrity of all Google Cloud Key Vault hardware operations.
- **Gneiss firmware signing keys** and any other additional signing keys supported by the Titan BootROM and BootLoader are used by the Titan chip to support a secure boot chain where each software module, beginning with the BootROM, verifies the cryptographic signature of each subsequent software module. The actual keys used in the case of Google Cloud Key Vault are development keys that are widely available to Google developers. The upgrade scheme implemented in the Google Cloud Key Vault firmware does not rely on the secrecy of these signing keys, but instead relies on the integrity protection of the firmware update log to detect rogue updates.

The Gnesis card security architecture and possible threat vectors from attackers is depicted in the diagram below.

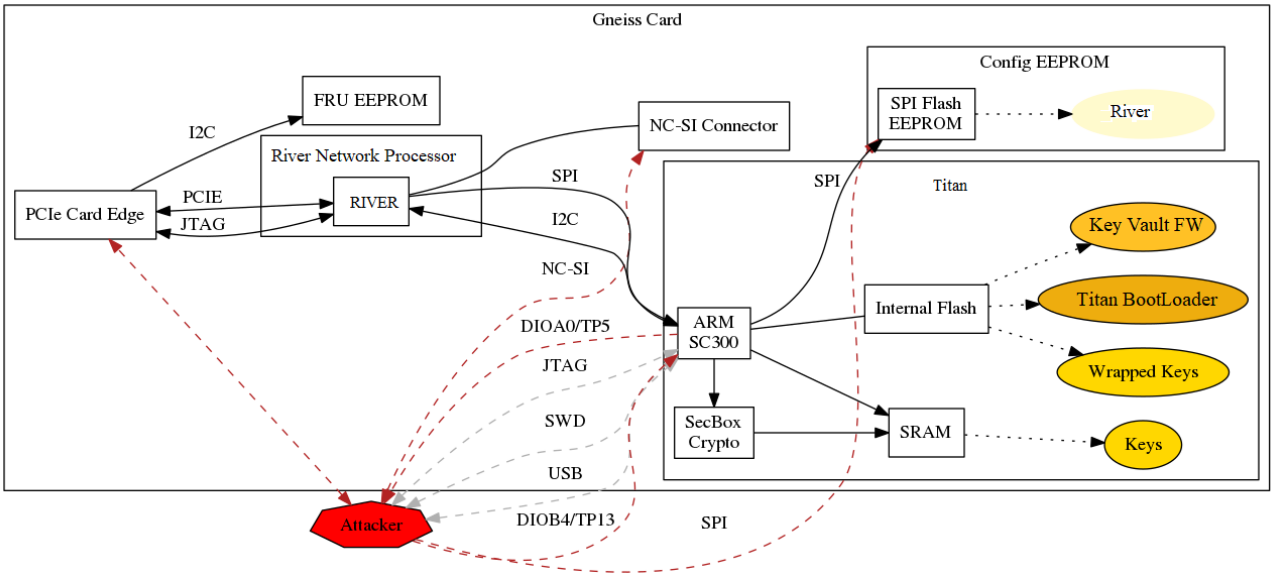


Figure 3: Gneiss card security architecture

Google Cloud Key Vault Firmware

The Google Cloud Key Vault software stack consists of three components: the BootROM, BootLoader, and Google Cloud Key Vault Firmware. The BootROM is baked into the Titan chip by design. While it is immutable, there are a handful of security-impacting fuses that can alter its behavior. The BootLoader shares a common ancestry to other Titan-based designs and is already in the open-source ChromiumOS project.[2] The attack surface of the BootROM and BootLoader is minimal, while the Google Cloud Key Vault firmware contains a large command handler. There are two flash memory banks within the Titan chip, and these are used to hold two redundant copies of the firmware (for fault tolerance during updates).

[2] Google Cloud Key Vault BootLoader source code

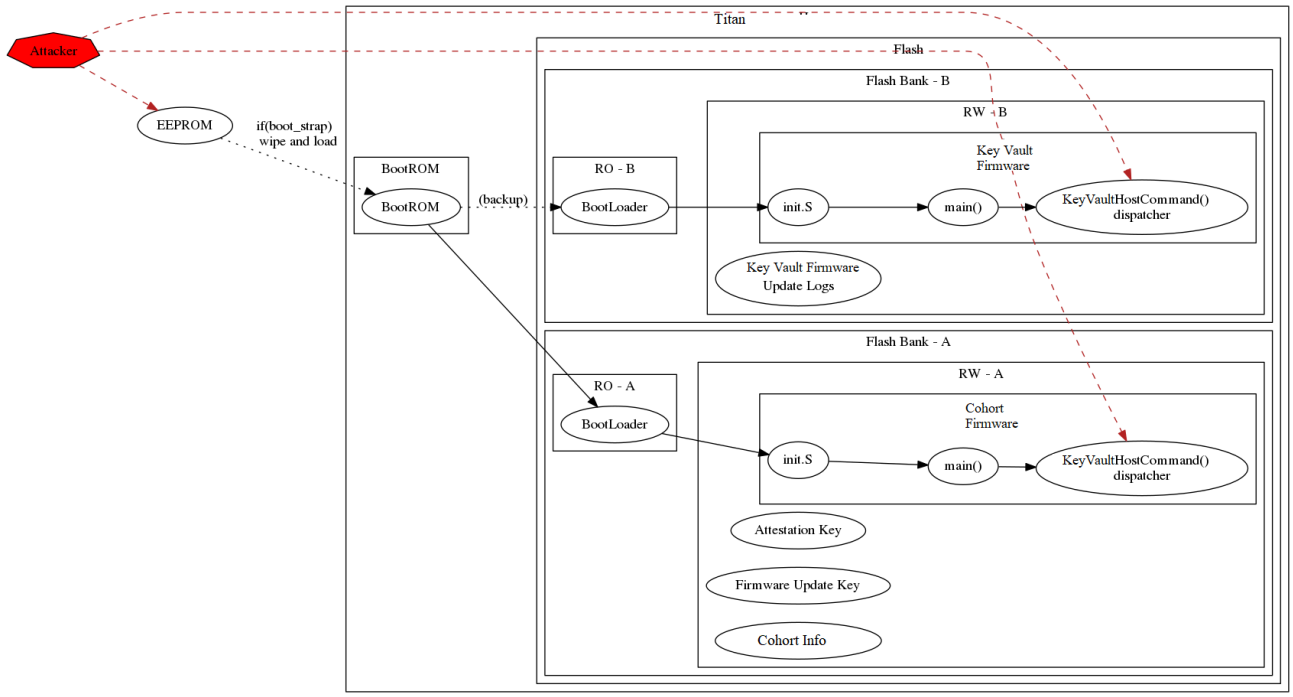


Figure 4: Google Cloud Key Vault software stack architecture

The Google Cloud Key Vault firmware itself, was custom written for this project. It exposes commands for diagnostics, cohort provisioning, and most importantly, password challenge verification and brute force countermeasures. The main brute force countermeasure consists of storing a per-user monotonic counter that is used to prevent more decryption requests than permitted by the Android client (currently 10). These per-user counters are externally stored in an encrypted Merkle tree format for efficiency. The tree itself is protected from rollback by a locally stored monotonic counter.

Secure Hardware Provisioning Process

This process is where the Gneiss cards are sent for provisioning before deployment. The initial Google Cloud Key Vault firmware is loaded and cohorts are configured. This happens at a secure facility in the HSM provisioning room before the Gneiss cards are deployed to data centers and can receive customer data. While the provisioning process was not fully documented at the time of assessment, it does rely on multiple trusted operators to ensure it happens correctly.

Secure Scrap Process

The Titan chip, once provisioned in a cohort, contains all the secrets needed to decrypt the user's backup encryption key. As such, it must be safely disposed of when the Gneiss card fails in an irrecoverable manner. While the process for this was not documented at the time of assessment, the outline of the planned process is as follows:

1. The failed Gneiss card is removed from the production server and placed in a locked cabinet to await destruction.
2. A suitably trained data center technician removes the card from the storage cabinet, and using a grommet press (depicted below), physically crushes the Titan chip. This destroys the chip itself and ensures that no key recovery is possible.
3. The Gneiss card is then placed in the regular media destruction flow. This process involves a mechanical shredder; however, its granularity is considered too large to definitively assure the destruction of the tiny Titan chip (hence the need for the previous steps).

Figure 5: Grommet press

Prober health monitoring utility

The *prober* utility is responsible for the continuous monitoring of the Google Cloud Key Vault devices. It runs at intervals of a few minutes and will cryptographically verify the integrity of each cohort, including the firmware update logs, and individual public keys. Any device that goes offline for any amount of time will be detected and an alert sent. If the downtime is unplanned, then the operators can take the cohort out of serving mode by pushing new public keys to all Android clients. The affected cohort can then be left in "drain mode" for a period of time to allow for any necessary restore operations to happen as needed until such time as the cohort is erased entirely.

Vault service

This is a OnePlatform service that manages the Vault and allows the enrollment of recovery and application keys. The user's phone will communicate to the Vault service anytime the user unlocks their phone or changes their unlock secret. The Vault service will provide the public key of a "cohort" of five (5) Titan chips that are located on the Genesis cards. The Vault service also communicates with the Google Cloud Key Vault service running on each of these hosts. The Vault verifies the user's local secret knowledge factor (LSKF) and provides the encrypted keychain content, such as the recovery key and application keys. The Vault service is implemented and deployed via OnePlatform and exposes the following methods: CreateVault, OpenVault and ListVault.

Android changes

The main goal is to allow users to backup their data in such a way that Google is unable to access it by encrypting the user's data with a key that Google never has direct access to. This backup encryption key will be stored only in secure hardware on the phone and synced periodically with secure hardware hosted by Google. The backup encryption key is encrypted with a key derived from the user's LSKF. The new APIs in the Android Framework that allow a user Keychain to be synced to Google in an encrypted form and protected by the LSKF and Google Cloud Key Vault cohort keys were reviewed during this security assessment. The high-level Android design of Google Cloud Key Vault is shown below.

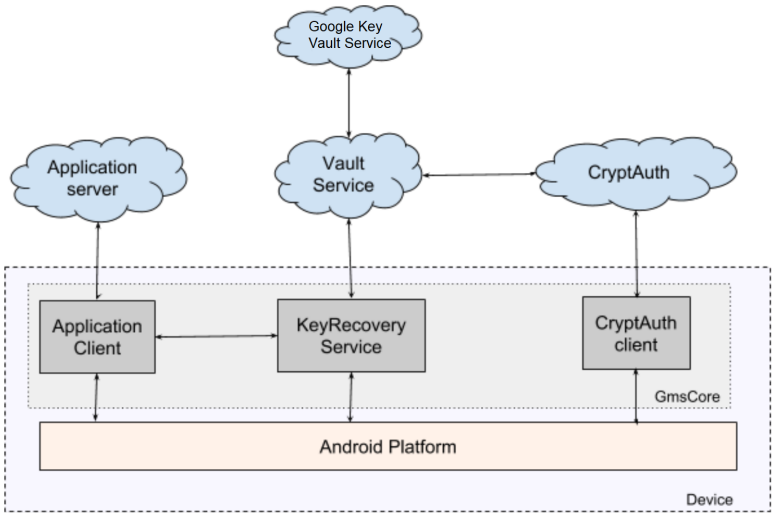


Figure 6: High-level Android design

GmsCore

This is the Google Mobile Services component. It is the non-open-source component on Android devices that enabled connections to the Google back-end services such as PlayStore. It is cryptographically signed by Google and verified by the Android operating system as part of the normal boot-time process. A compromise of this endpoint would be sufficient to capture backup data at the source, or to subtly cripple the integrity of the backup/restore process to permit remote attacks on the backups (eg. set max_attempts in the vault parameters to 255).

Backup and Restore Keys

Google Cloud Key Vault public key. The public key shared by a Google Cloud Key Vault cohort (a group of 5 Google Cloud Key Vault chips located at different data centers). Since Google has several cohorts, there exist several of these keys that GmsCore can randomly choose from (once) to back up their recovery key. These public keys are listed in an XML file which is signed by a trusted Google certificate and obtained during the initialization of the Android phone.

LSKF-bound encryption platform key. A key that is stored under two different aliases in the Android Framework KeyStore. This key contains a "purpose decrypt" alias that can only be used when the screen is unlocked (auth required: true) and a "purpose encrypt" alias that can be used when the screen is locked (auth required: false). This key is generated by the Android Framework RecoveryController and used to wrap the application key. It is only relevant as an "encryption at rest" mechanism and not strictly relevant to the Google Cloud Key Vault protocol.

Application key (or Secondary key). This is a 256-bit AES GCM key created by the GmsCore Backup Module (`generateAndStoreKey`) and is associated to an alias. It is encrypted at rest with the Platform Key. Here "Application" refers to the only currently available application of Google Cloud Key Vault (backups) and not to the applications available on the Google Play Store. It is the key used to encrypt/wrap tertiary keys for the backup application. For future applications it might be used for different purposes. It is stored on the Google servers (Spanner) after having been encrypted by the Recovery Key. Ultimately this is the key that our "application" is backing up and we recover it via the recovery key.

Recovery key. Created by the Android Framework RecoveryController when the LSKF changes. It is associated with a `vault_params` metadata blob (also re-generated when an LSKF changes). It is used to wrap the Application keys. This is the key that we recover via the Google Cloud Key Vault protocol. In the first part of that protocol (key backup) the recovery key is wrapped twice (two layers of encryption) by hashes of the LSKF and encrypted to the Google Cloud Key Vault's public key. Ultimately, in the recovery part of the protocol, the Google Cloud Key Vault will remove one of the

encryption layers on this recovery key for us, and we will remove the second layer via our LSKF.

Vault parameters. A blob generated by the Android phone when the LSKF changes, it contains metadata about the current user's Android id and Instance id, the choice of Google Cloud Key Vault cohort (their public key), a random 64-bit id, and the maximum number of attempts allowed (hardcoded to 10). This blob is used to uniquely authenticate users and their current LSKF (associated with a number of bad recovery attempts).

local_lskf_hash. A SHA-256 hash of a public salt and the user's LSKF.

key_claimant. A symmetric session key encapsulated with a Titan public key and used to encrypt the response from the Titan chip to the phone.

Tertiary key. Key used to encrypt actual Google Play Store application backup. Backups can be blobs (dolly backup) or key-values pairs. These tertiary keys are wrapped with the secondary keys (application keys) and stored along encrypted backups.

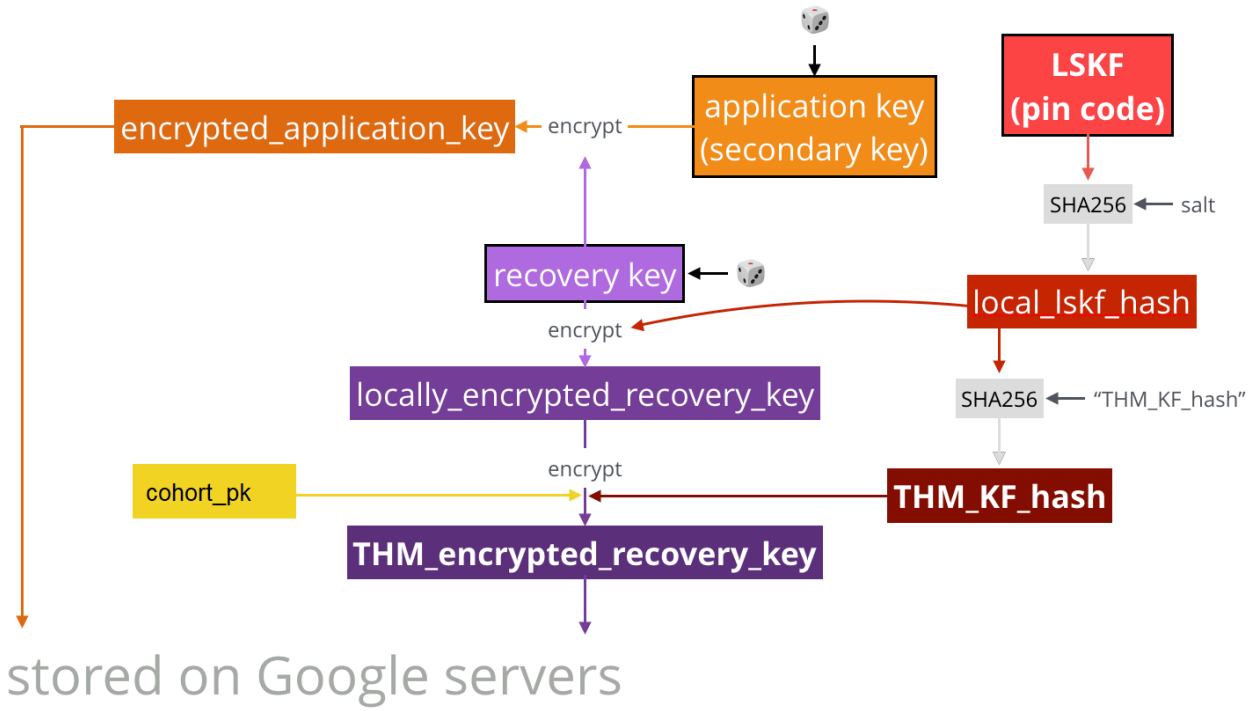


Figure 7: The backup process of the Android client. Authenticated Data fields are missing from the diagram.

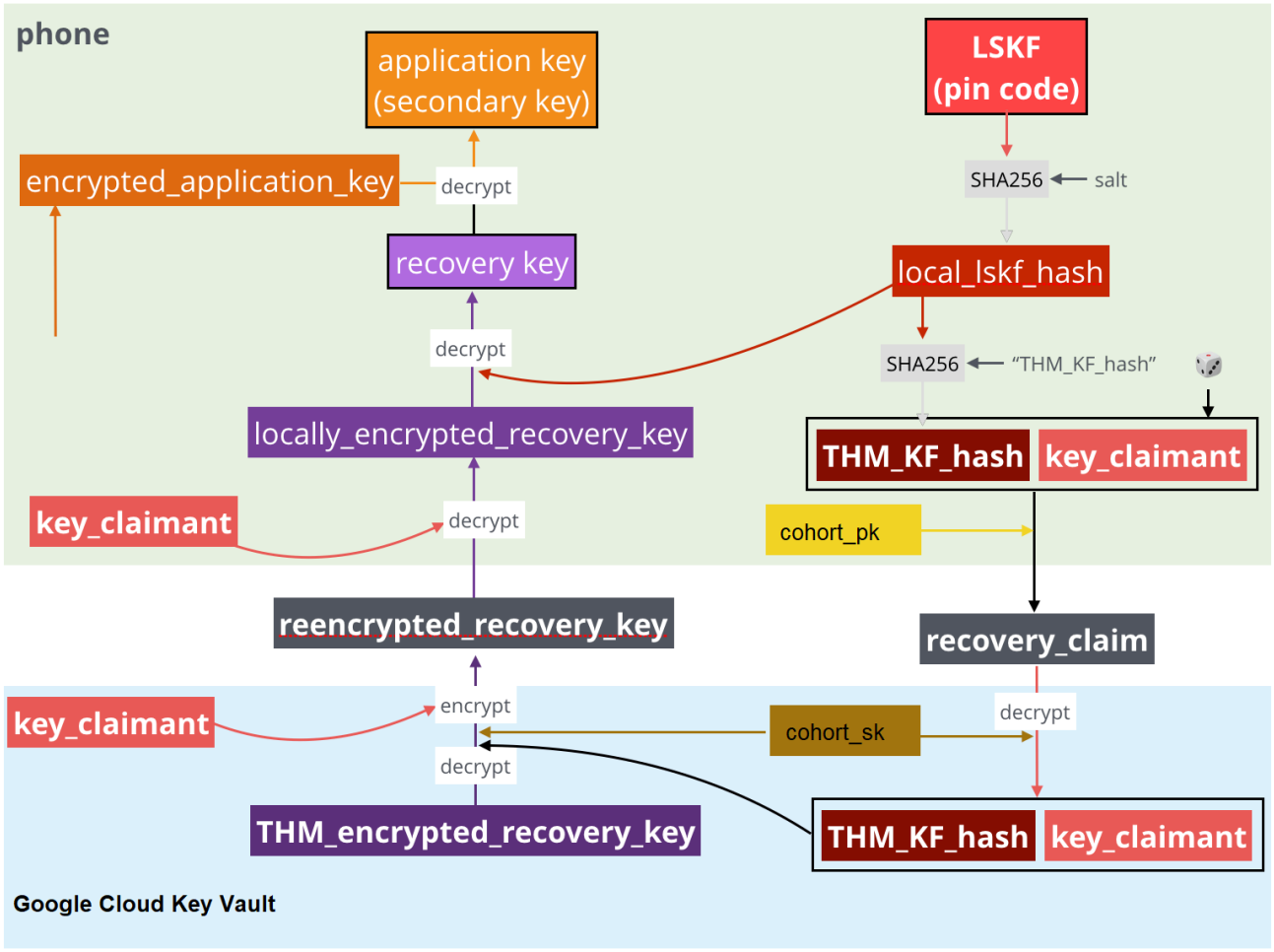


Figure 8: The recovery process of the Android client talking to the Titan chip. Authenticated Data fields are missing from the diagram.

Assessment Summary

Target Metadata

Name	Android Encrypted Backup-Restore
Platforms	Custom Google Cloud Key Vault firmware Android

Engagement Data

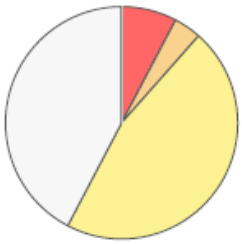
Type	Cryptography and embedded device security assessment.
Method	White-box
Dates	2018-07-09 to 2018-08-03
Consultants	8
Level of effort	94 person days

Targets

Cryptographic mechanisms and protocols	Google Cloud Key Vault Service Google Cloud Key Vault Android Client Google Cloud Key Vault and Titan Hardware Google Cloud Key Vault and Titan Firmware Google Cloud Key Vault service and systems deployment
Design and implementation	Google Cloud Key Vault and Titan Hardware Google Cloud Key Vault and Titan Firmware
Deployment Review	Google Data Center

Finding Breakdown

Critical Risk issues	2
High Risk issues	0
Medium Risk issues	1
Low Risk issues	12
Informational issues	11
Total issues	**26**



Category Breakdown

Auditing and Logging	3
Configuration	3
Cryptography	9
Data Exposure	4
Data Validation	5
Error Reporting	2

Component Breakdown

GmsCore	11
Key Vault Firmware	13
Key Vault Hardware	1
Key Vault Service	1

Key

Critical High Medium Low Informational

Finding Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines. As mentioned on Page 3 under Key Findings, several low severity issues were reported that do not pose an immediate threat to the system or to the users. These low severity items have been reviewed and are being prioritized accordingly.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise. Vulnerabilities that can easily be exploited by either external or internal attackers to gain access to the user data or backup encryption keys are classified as critical (for example, a vulnerable that reveals arbitrary memory of the Google Cloud Key Vault device, or allows brute forcing of the LSKF).
High	Implies an immediate threat of system compromise, or an easily accessible threat that does not easily compromise the user's backup encryption keys.
Medium	A difficult to exploit vulnerability, or easy compromise of a small portion of the system.
Low	Implies a relatively minor threat to the system.
Informational	No immediate threat to the system. May provide suggestions for improvement, functional issues with the code, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible actions, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCCGroup categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Finding Definitions on page 16.

GmsCore

Title	Status	ID	Risk
Encrypted Chunks and Chunk Ordering Share AES Key	Reported	013	Low
Key-Based Fingerprint Mixing For Content-Defined Chunking Is Not Effective	Reported	018	Low
Tertiary Keys Are Interchangeable Due To Lack Of Metadata Authentication	Reported	023	Low
Secondary Keys Are Interchangeable Due To Lack Of Metadata Authentication	Reported	026	Low
Google Can Force Applications To Restore From Arbitrary Backup Data	Reported	027	Low
Key Reuse Across Different Algorithms	Reported	015	Informational
Backup Age Is Not Validated	Reported	019	Informational
Negative Array Size Exception Not Caught	Reported	020	Informational
Chunk Metadata Contains Unauthenticated Fields	Reported	024	Informational
Adversary Can Actively Identify Identical Plaintext Chunks	Reported	025	Informational
Adversary Can Passively Identify Identical Plaintext Chunks	Reported	028	Informational

Key Vault Firmware

Title	Status	ID	Risk
Panic Function Can Leak Information	Fixed	004	Critical
Unlimited Recovery Attempts Are Permitted By Titan Firmware	Fixed	008	Critical
`panic()` is Not Resistant to Fault Injection Attacks	Reported	002	Low
Inconsistent Use of `zeroMemory()` to Erase Sensitive Data from the Stack	Reported	003	Low
Signature Verification Primitive Lacks Some Range Checks	Fixed	006	Low
Public Curve Point Validation Is Incomplete	Fixed	007	Low
Firmware Does Not Verify Fuse Provisioning	Updated	010	Low
Firmware Code Signing Manifest Can Be Hardened	Reported	014	Low
Console Debug Logging is Enabled	Updated	001	Informational
Anti-Tamper Feature Does Not Clear Sensitive Keys from RAM	Updated	005	Informational
No Runtime Characterization of TRNG	Reported	009	Informational
Guard Pages Are Not Used to Protected Sensitive Items in RAM	Reported	016	Informational
No Rate Limiting in `FOLSOM_CHALLENGE_FIRMWARE` Command	Reported	017	Informational

Key Vault Hardware

Title	Status	ID	Risk

Google Cloud Key Vault Titan Chip Fusing Has Token-locked TESTMODE Enabled	Reported	022	Low

Key Vault Service

Title	Status	ID	Risk
Prober Tool is Not Deployed	Reported	021	Medium