Usable Key Management Pt. 2

Section 1—Activity 1

How long did it take you to complete this activity?

10 Minutes

What steps did you take to complete the activity?

- Include details about anything you attempted that ultimately did not work.
- Include screenshots if you think that would be helpful.

I exported my gpg private key, copied it over to hydra, installed the key, and enabled commit signing.

```
gpg --export-secret-keys --armor YOUR_KEY_ID > private-key.asc
```

Identify the tools you used.

- Also, include details about any tools you ultimately abandoned.
    - Describe what went well with these tools and what was challenging.

I used the same tools as part 1, just the gpg & git command line tools. Both tools were easy to use and I had no issues.

Describe what information sources you used and how helpful (or unhelpful) they were.

- Why did you need to use them?
- Why were they helpful (or unhelpful)?

I needed to look up the correct set of flags to export my private key, which I found at https://gist.github.com/luckygoswami/3f7cd9a2a7787314d44290941f0665f3.

To complete this activity, you could either copy your private key to the new machine or generate a new key pair for the new machine. Which approach did you choose? Why?

Answer the following questions:

- What were the easiest one or two steps in setting up Git commit signing on a second device? Why were they the easiest?
    - The easiest step was exporting my gpg private key. This was the easiest as I copied a command.
- What were the hardest one or two steps in setting up Git commit signing on a second device? Why were they the hardest?
    - The hardest step was copying my gpg key over because UTK requires me to use a VPN in order to be able to SSH onto a computer.
- What would you change about the process for setting up Git commit signing on a second device, if anything?
    - Nothing needs to change in my opinion.

Answer the after-scenario questionnaire (ASQ) by indicating how much you agree with the following statements on a scale of 1 (strongly disagree) to 7 (strongly agree).

- Overall, I am satisfied with the ease of setting up and using commit signing on a second device.
    - 7
- Overall, I am satisfied with the amount of time it took to set up and use Git commit signing on a second device.
    - 7
- Overall, I am satisfied with the support information (online help, messages, documentation) I found when setting up and using commit signing on a second device.
    - 7

Provide any other feedback you have about this activity or Git commit signing on a second device.

I have no other feedback.

Section 2—Activity 2

Did you identify any commits that you think are problematic?

- If so, what was problematic about that commit?
- Provide these details for each commit you believe is problematic.

On either end of the commit list we have 2 commits that are both years apart from the rest of the commits. I find this suspicious, and would suspect that these commits have had their dates modified. Within the bulk of the commits we have most commits showing as verified and properly signed, mixed in with a few unverified commits, and some that are not signed. The commits from Dr. Ruoti seem to be the most problematic as there is never a signing key associated with his commits. Additionally, individual commits from both John and Abubakar are not signed, which does not instill confidence that these commits are legitimate. Ultimately, it would be up to the developer choosing to consume this library to enact their own policy according to the level of risk they are willing to take in 3rd party code.

How long did it take you to complete this activity?

10 minutes.

What steps did you take to complete the activity?

- Include details about anything you attempted that ultimately did not work.
- Include screenshots if you think that would be helpful.

I cloned the repo, added the developers keys to my local store of keys, and then ran `git log --show-signature`.

Identify the tools you used.

- Also, include details about any tools you ultimately abandoned.
- Describe what went well with these tools and what was challenging.

I used the git cli.

Describe what information sources you used and how helpful (or unhelpful) they were.

- Why did you need to use them?
- Why were they helpful (or unhelpful)?

I did not need any external sources.

-Answer the following questions:

- What were the easiest one or two steps in analyzing the Git commit history? Why were they the easiest?
    - The easiest was cloning the git repo, this is something I do almost daily so it is easy.
- What were the hardest one or two steps in analyzing the Git commit history? Why were they the hardest?
    - The hardest part was adding the developers keys to my local store, because I initially forgot where those keys live.
- What would you change about the process of analyzing the Git commit history, if anything?
    - Nothing.

Answer the after-scenario questionnaire (ASQ) by indicating how much you agree with the following statements on a scale of 1 (strongly disagree) to 7 (strongly agree).

- Overall, I am satisfied with the ease of analyzing the Git commit history.
    - 7
- Overall, I am satisfied with the amount of time it took to analyzing the Git commit history.
    - 7
- Overall, I am satisfied with the support information (online help, messages, documentation) I found when analyzing the Git commit history.
    - 7

Provide any other feedback you have about this activity or analyzing Git commit histories.

None.

Section 3—Semester-long reflection

For this section, please reflect on your experience using Git commit signing in the second half of the semester.

Git commit signing is something that I have been doing for a few years now so the entire process was familiar to me. I think it's a good experience, especially with the guidance from your choice of git hosting provider guiding you along the way.

Where did you store your private key? How was it secured?

My private key was stored password protected on my laptop.

Did you change anything about how you signed commits throughout the second half of the semester? If so, what changes did you make? Why did you make those changes?

I did not, I chose the correct method the first time around.

Do you plan to start signing your Git commits outside of this class? Why or why not?

Yes, I will continue to sign my commits outside of the class. I like to provide my collaborators with the confidence that I am truly the one creating commits.

Answer the system usability scale (SUS) questions by indicating how much you agree with the following statements on a scale of 1 (strongly agree) to 5 (strongly disagree).

- I think that I would have no problem using Git commit signing frequently.
    - 5
- I found using Git commit signing unnecessarily complex.
    - 1
- I thought that using Git commit signing was easy.
    - 5
- I think that I would need the support of a technical support staff to use Git commit signing in the future.
    - 1
- I found the various functions for using Git commit signing to be well-integrated.
    - 5
- I thought there was too much inconsistency in using Git commit signing.
    - 1
- I would imagine that most people would learn to use Git commit signing very quickly.
    - 4
- I found using Git commit signing to be very cumbersome.
    - 1
- I felt very confident using Git commit signing.
    - 5
- I needed to learn a lot of things before I could get going with using Git commit signing.
    - 1

Answer the following questions:

- What were the easiest steps to set up Git commit signing, sign a commit, or verify a commit? Why were they the easiest?
  - The easiest step was to verify a commit, which can be accomplished using `git log --show-signature`. This is a fairly simple command to remember, and it can be set as a default using your git config.
- What were the hardest one or two steps in setting up Git commit signing, signing a commit, or verifying a commit? Why were they the hardest?
  - The hardest step was remembering to configure git to enable automatic commit signing. I found this the hardest as the git cli flags for config options are not intuitive.
- What would you change about the setup process, if anything?
  - If more steps could be automated that would be great, likely during an OS install or something similar.

How did the usability of setting up git commit signing compare to the usability of using it throughout the second half of the semester?

Using it throughout the second half of the semester was easy, and I didn't any change in complexity.

Provide any other feedback you have about Git commit signing.

None.

Section 4—Thought exercises

In this section, you will complete several thought exercises. You will receive full credit for whatever you put down. I'm not looking for you to get the "right" answer, but rather for you to share your thoughts based on what you've learned this semester and your experiences using Git commit signing.

Are there any security benefits or drawbacks to using Git commit signing as compared to unsigned commits? If so, what are they?

Assuming (as with all cryptography) that the secrecy of the keys is kept fully secure git commit signing guarantees that the actual author associated with the commit was the one that wrote it. For safety/security critical code bases this is an extremely important fact, and therefore signing provides a benefit in these situations. Additionally, in larger open source projects git commit signing can help to instill confidence that a contributor is really who they say they are, assuming a network of trust is in place.

To use Git commit signing on multiple devices, you could either synchronize your existing signing key to the new machine or generate a new signing key for that machine. Each of these approaches has different potential security implications. What do you think the security benefits or drawbacks of each approach are?

Copying keys between machines exposes the potential for a broader attack surface, as you are sharing a single key. This means that if any of the copies is exposed, they are all exposed. However, this does make sharing your public keys with others much easier, as they only have to have a single key for you.

On the other hand, having individual keys per machine reduces the risk of being wholly compromised from a single attack, but it greatly increases the complexity of managing public keys.

If you were to lose your Git signing private key, what would you need to do to continue signing commits? Are there any security concerns with your proposed workflow? If so, how could they be addressed (you can answer that you don't know)?

You would need to generate a new key, that would then be distributed to all relevant parties. At the same time you should also ensure that your other key is "forgotten" so that if someone else manages to recover it, they cannot use it to continue impersonating you. The security concern is that a bad actor could just as easily compromise some other piece of your security chain, and use that to

convince your network that their new key is yours.

If you were to have your Git signing private key stolen, what would you need to do to ensure the security of your repository? Are there any security concerns with your proposed workflow? If so, how could they be addressed (you can answer that you don't know)?

You would need to revoke the use of that key as soon as possible. The security concern is if that key was used elsewhere you would need to ensure its use is blocked on every platform.

Consider an open source project that's widely used, and attackers would like to get their code into this repository. What might attackers try to do to compromise that repository? What practices would you recommend that developers of this repo take to ensure that no malicious commits are added to their codebase?

As an attacker I would choose to play the long game where I initially submit good-faith pull requests to build trust with the maintainers of a project. After this trust had been established I would then take some piece of code that I was responsible for and rewrite it to insert my malicious code, knowing that I have the most authority over this area of the code.

Date: Mon Dec 1 17:19:09 2025
Author: Jackson Mowry
Created: 2025-12-01 Mon 18:18