Signal MoonMoon Technologies Release Summary

Team members

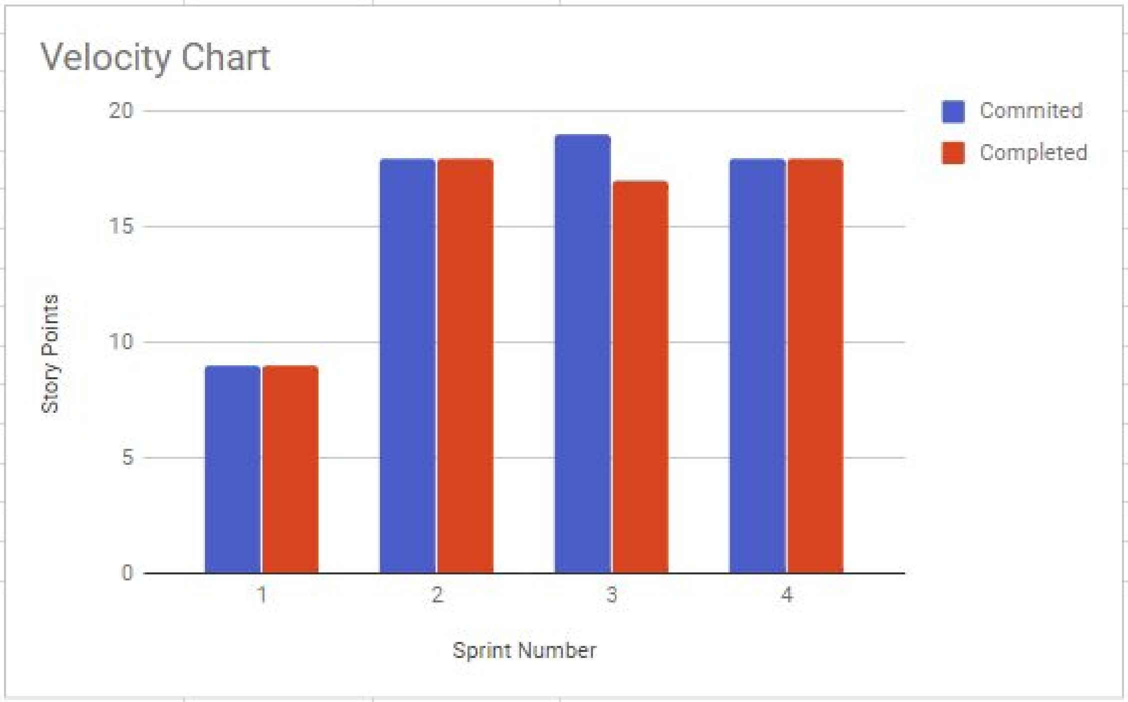
Name and Student id		GitHub id	Number of story points that member was an **author** on.
Jacob Gagne	**40003704**	jacobrs	Sprint 3: 5 Sprint 4: 8
Benjamin Barault	40003661	benrs	Sprint 3: 3 Sprint 4: 3
Lori Dalkin	27738293	lori-dalkin	Sprint 3: 5 Sprint 4: 5
Claudia Della Serra	27677048	claudds	Sprint 3: 3 Sprint 4: 8 +2 bugs
Bryce Drewery Schoeler	27283199	BDSchoeler	Sprint 3: 8 Sprint 4: 5
Samantha Kerr	40007328	samantha-kerr	Sprint 3: 8 Sprint 4: 8

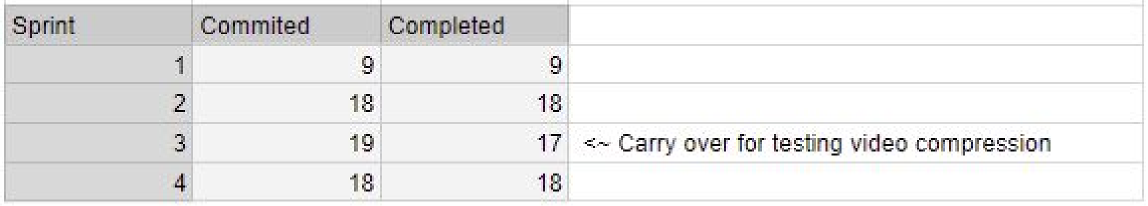
Each group member is responsible for counting their own story points. It is the group leader's duty and responsibility to make they are accurate. The team lead should be listed first.

Velocity

Only stories that have stakeholder sign off, demo steps, and tests are counted. We had carry over from sprint 3, so we added 2 story points worth of video compression testing to sprint 4.

Total: 7 stories, 35 points over 4 weeks





Sprint	Commited	Completed	
1	9	9	
2	18	18	
3	19	17	<~ Carry over for testing video compression
4	18	18	

Plan up to next release

Release 2
Sprint 3 (4 stories, 19 points) (carry over of 1 story to sprint 4)
Sprint 4 (4 stories, 16 points)

Release 3
Sprint 5 (5 stories, 15 points)
Sprint 6 (3 stories, 21 points)

Overall Arch and Class diagram

Please describe any changes in your arch!

Infrastructure

SiliCompressor

The SiliCompressor library provides image and video compression methods for Android applications. This library was selected for our video compression feature. The library's *compressVideo()* method that was implemented required as parameters the URI of the video to be compressed and a destination directory path, and returned the file path of the new, compressed video. The message attachment is then updated with the video's new filepath, and the compressed video is sent to the recipient. The library allows compressed videos to maintain their quality, and doesn't require decompression on the recipient's end; this was especially useful for Signal, as it would have only been possible for us to implement decompression if the recipient was a user of the Signal app.

Name Conventions

List your naming conventions or just provide a link to the standard ones used online.

1. Signal Code Style Conventions
2. Java naming conventions

Code

Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with clickable GitHub link	Purpose (1 line description)
Signal-Android/src/org/thoughtcrime/securesms/ConversationItem.java	Item representing the messages in the application; added in icons for marking as unread & pinning, as well as implemented a filter & highlighting for searching here.
Signal-Android/src/org/thoughtcrime/securesms/PinnedMessagesListActivity.java	New activity for pinned messages, shows only those messages which have been pinned for a given conversation
Signal-Android/src/org/thoughtcrime/securesms/mms/MediaConstraints.java	Class defining media constraints for MMS messages; add a method to call the video compression method and return compressed video filepath.

Signal-Android/src/org/thoughtcrime/securesms/ConversationFragment.java	Fragment that controls most UI components for a conversation. Includes code for pinning, unread, and search term features.
Signal-Android/src/org/thoughtcrime/securesms/FingerprintAuthenticationHandler.java	New class that handles fingerprint authentication. Generates keys, handles successful authentication attempts.

Testing

Each story needs at least a tests before it is complete.
If some class/methods are missing unit tests, please describe why and how you are checking their quality. Please describe any unusually aspects of your testing approach.

List the **3** most important **unit test**s with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
Signal-Android/test/unitTest/java/org/thoughtcrime/securesms/database/SmsDatabaseTest.java	This, along with the MmsDatabaseTest, is unit testing the added fields to track messages that are pinned and marked as unread
Signal-Android/test/androidTest/java/org/thoughtcrime/securesms/FingerprintAuthenticationTest.java	These tests are run on the emulator and are unit testing different crypto components in order to confirm they are using the right providers and algorithms.
Signal-Android/test/unitTest/java/org/thoughtcrime/securesms/ConversationItemTest.java	Tests testing multiple cases for searching and highlighting message text.

List the **3** most important **UI end to end (espresso) tests** with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
Signal-Android/test/androidTest/java/org/thoughtcrime/securesms/GetFakeVerifiedTest.java	Test allows us to bypass the verification steps which was previously hindering our ability to perform espresso tests, it now does so automatically without user input.
Signal-Android/test/androidTest/java/org/thoughtcrime/securesms/espressoTests/MarkMessageAsUnreadTest.java	UI Test for the marking messages as unread story which we completed in Sprint 3.

Signal-Android/test/androidTest/java/org/thoughtcrime/securesms/util/ConversationActions.java	A bunch of different reusable UI Test actions that we use in our UI Tests. These functions are abstracted so that it's easier for us to re-use common UI functionality (like going to a specific conversation).

Finished SHORT Story summaries

Order your summaries by risk and priority. They should have the following form.

Points: 8, Priority: 4/4, Risk: medium
Story #40 Search Through Convo
Sprint #4
Feature: Search through conversation

Added a feature to allow searching through conversations. Feature allows the user to enter a search term in the conversation search bar, accessed through a menu icon; this search bar takes advantage of the SearchView widget native to Android. Since messages are encrypted, filtering is done at the RecyclerView level in order to access the displayed body, and filter through the text in there. Only those messages containing the search term are then displayed on the screen, and the search term is highlighted within each message for easier viewing.

Points: 8, Priority: 4/4, Risk: medium
Story #34 Video Compression
Sprint #3
Feature: Video compression

Added a feature to compress video file attachments when sending a message. Compression is done using the SiliCompressor library, which takes a file path and destination directory and returns a new path to the compressed video. Compression is done automatically when sending a video in a message. There's minimal loss of quality in the compressed video; it is able to be viewed by the recipient without the need for decompression.

Points: 5, Priority: 3/4, Risk: medium
Story #37 Fingerprint Authentication
Sprint #4
Feature: Fingerprint authentication

Added a feature to enable a phone's built in fingerprint sensor as a mode of authentication when accessing the application. The feature introduces a new setting toggle in privacy settings that becomes available when Passphrase authentication is enabled/set and if the phone's hardware supports it. When fingerprint authentication is enabled, a fingerprint icon appears on the passphrase prompt page to indicate a fingerprint scan can unlock the app. It works by using RSA where the private key is only usable when the user identifies themselves with their fingerprint, and the public key is used whenever the user sets the password to encrypt it.

Points: 5, Priority: 3/4, Risk: High
Story #38 Pin messages
Sprint #3
Feature: Pin messages

Added a feature to mark multiple messages as pinned in order to then access all pinned messages in a new activity. This feature introduces a new pin and unpin menu icon that toggles the pins status of a message. Pinning a message also adds a small pin icon to the bottom of the message. Users can now access a Pinned Messages activity per thread, where all pinned messages are available for viewing. This feature was divided into two parts. Lori handled the Conversation activity elements while Jacob created the new Pinned Messages activity.

Points: 3, Priority: 2/4, Risk: medium
Story #36 Mark a message as unread
Sprint #3
Feature: Mark as unread

Added feature to mark a message as unread in order to set a reminder on the conversation for users. Claudia implemented all additions to ConversationFragment, messaging databases, and alert views as well as added new notifications to the UI. This feature closes the conversation and sets a notification on that conversation. Once the conversation is reopened, notifications disappear, but messages that were marked as unread remain with an read reminder.

Points: 2, Priority: 2/4, Risk: medium
Story #82 Remember Sending Message State
Sprint #4
Feature: Remember Sending Message State

Used Shared Preferences to store if the user manually selected a specific messaging state. Whenever the user revisits a specific conversation it will use the recipient's short name to check if they have a preferred messaging state, if it's found then it will set it. Also added UI tests to test this, no unit tests because all the core logic is in an extremely coupled activity (ConversationActivity).

Points: 1, Priority: 4/4, Risk: medium
Story #77 Implement Espresso Tests and Enhancing Fake Verification

Sprint #4
Feature: **Enhance Espresso** and **Espresso Tests**

Enhanced the fake verification method, now it automatically accepts all the permission dialogs without requiring user interaction. Also added new libraries, one of them being the UI Automator library which more easily lets us interact with views that are normally hard to interact with using Espresso. On top of that we added helpers static methods that abstract ui testing tasks that are usually repeated, e.g entering a conversation. Finally after adding all these enhancements UI tests were adding for pinning a message and marking a message as unread as this could not be done in Sprint 3 because the UI tests were only fixed near the end.

Story #56 Fix Espresso Tests
Sprint #3
Feature: Fix Espresso

Fixed the gradle build, previously it was causing our androidTests directory to not properly be built and for that reason we could not add espresso tests. Also got tests that were not previously compiling to compile. Those tests fail but were added by previous developers that worked on Signal and are not UI tests. After fixing the gradle build a method to bypass verification was added in the form of an espresso test. After running the test your phone or emulator will be verified with fake information.

CI stages and notification

Jenkins CI is now split into two different jobs. Each job is responsible for a unique gradle wrapper command. The build job simply ensures all dependencies are available at compile time and the code does not contain any compilation errors. The second job is a test job that runs the test gradle command. This will bring attention to failing tests by running the test suite on the build system and reporting invalid exit codes.