Versatile Web App Implementation for NFC NDEF Tags

Andrew Simon, University of North Florida, CIS 6167: Internet of Things

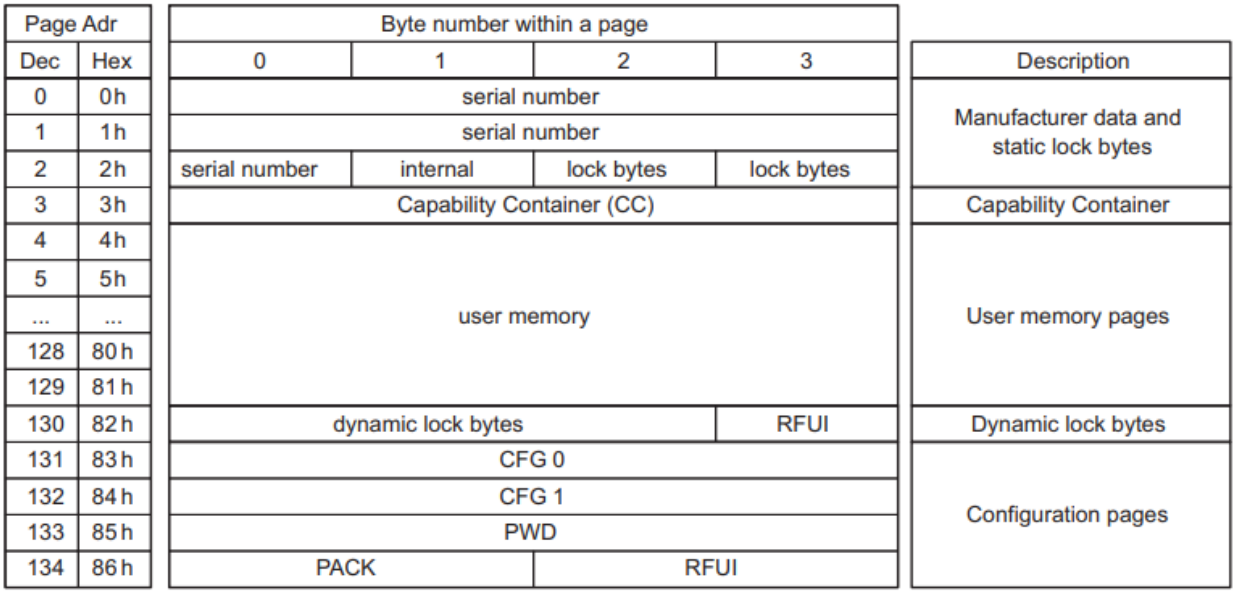
My technology of focus this semester was Near Field Communication (NFC) tags. The specific model of tags I experimented with was the NTAG 215 tags produced by NXP Semiconductors. This model of NFC tag is a Type 2 tag, following the ISO/IEC 14443A standard. This technology allows for an NFC-reading-capable device to interact with the tags within an approximate 10-centimeter distance. The focus of my experimentation was learning to write to these tags, read/understand the memory data of the technology, and consider methods to build a web-based app that could perform both of those actions effectively.

NFC tags are categorized into five tag "type" groupings, increasing in sophistication. Type 1/2 tags are similar in that they both follow the ISO/IEC 14443A standard, have read and write capabilities, and can be configured into a read-only state. With both having a lower starting memory range (96/48 respectively), these tags are often used for single-use applications, such as storing a URL or connection data. Type 2 tags are the most commonly used in the industry as they are incredibly cost-effective for the tasks they are being used for. Type 3 tags are exclusive to Sony and provide a larger memory (2k bytes) and a higher data rate (212kbits/s). This raises the cost of the product as well as decreases its diversity of suppliers. Type 4 tags increase both factors, and the overall security, and Type 5 tags are the newest, most sophisticated model, working on an entirely different standard (ISO 16693) [2].

The NXP NTAG Type 2 tags worked with came with 504 bytes of memory storage. This data in this memory was able to be locked in several increments. The lock bits, detailed further in this paper, allowed for specific sections of the memory to be configured into a read-only state. These tags also support 32-bit password protection working off AES encryption.

During experimentation, I used the NXP TagInfo mobile app to read the full memory organization of the tags and NFC Tools to practice writing data to the tags. NXP TagInfo provides users on Android devices to view the entire memory organization of the tag and supports access to every page in the organization for analysis. This functionality was used to check if my lock bits were correctly selected for configuration and show precisely how much memory I used with my stored data and where it was located. The NFC Tools app allowed me to edit the tag's pages once the initial write commands were set. This feature allowed me to tweak which bits of memory I wanted to make read-only and create a 32-bit password for security.

These tags use the NFC Data Exchange Format (NDEF) for communication between the reader and the tags themselves. This format outlines the actions that much be in place when encoding data on the tags or the data being exchanged between the reader and the receiver.



Page Adr		Byte number within a page				
Dec	Hex	0	1	2	3	Description
0	0h	serial number				Manufacturer data and static lock bytes
1	1h	serial number				
2	2h	serial number	internal	lock bytes	lock bytes	
3	3h	Capability Container (CC)				Capability Container
4	4h	user memory				User memory pages
5	5h					
...	...					
128	80 h					
129	81 h					
130	82 h	dynamic lock bytes			RFUI	Dynamic lock bytes
131	83 h	CFG 0				Configuration pages
132	84 h	CFG 1				
133	85 h	PWD				
134	86 h	PACK		RFUI		

Figure 1: NTAG 215 Memory Organization. Adapted from [1].

The memory organization for NTAG 215 tags consists of 135 pages, with each page split into 4 bytes. Type 2 tags require a 7-byte UID at the top of the memory followed by two check bytes. The next significant bytes are the static lock bytes on page 02h. The bytes can lock each specific page from 03h (CC) to 0Fh into a read-only state. There are so-called “block locking” bits in these bytes that completely freeze the locking configuration of the selected pages, preventing any future alteration of their read-only status. Another section of bytes to note are the dynamic lock bytes on page 82h. These locking bytes allow for a similar read-only configuration to be implemented toward pages 10h onwards. These three bytes are capable of locking 456 bytes of data on the tag [1].

The configuration pages at the bottom of the organization provide several useful features. The first byte in the page handles the ASCII mirroring functionality the tags support. This technique allows for the tags to paste a UID into a stored URL for backup/ease of use for the usage of the URL. The PWD byte stores the memory for the supported 32-bit password for memory access protection. This password’s location is stored in the AUTH0 byte, and a 16-bit password acknowledgment is stored in the PACK byte for proper password authentication. This security functionality is disabled by default but can be changed to block access to the configuration pages and any part of the user memory.

When looking to utilize this technology by building an app, the first area of focus was a mobile app. Working with Android Studio led to compatibility issues, as it was discovered that the platform does not support development on older Android OS versions. Unfortunately, the reader device being used supported Android Version 6, creating a roadblock for development. This hurdle led to an analysis of why the web app path was being chosen and how web development might be a more suitable format for the intended uses of NFC technology.

When thinking of the desired outcome of the app, it was noted that the goal was for customers or employees to be able to seamlessly interact with the technology in the most convenient way possible (The same can be said for the developers as well). Web applications can be launched at the scan of an NFC tag. Instead of having the intended users download a company-specific app to make efficient use of the tags, they could just scan a tag with the web app's URL encoded to its memory and get to work. This development method also covers the issue of dependence on OS versions. Designing a web app is not only version-independent but OS-independent entirely, allowing any NFC-capable reading device to interact with the app. While mobile apps can perform a bit faster and can be used offline, those benefits aren't necessary for the environment of intended use, and the benefits of web development vastly outweigh them. From a design standpoint, maintenance of a web app is far more cost and time efficient, as it doesn't require the frequent version updates and patching of mobile counterparts.

When deciding to focus on NFC technology, one primary topic was its comparative use to the QR Code. In many ways, these two technologies are direct competitors, and it intrigued me as to why one should be chosen over the other and if both technologies had a home in the overall ecosystem of the field. QR Codes are arguably more common in the everyday lives of most people. They can be accessed from any distance and are much easier to implement in a virtual setting. The downsides of this technology are its durability and ease of use. QR Codes are rendered completely useless if the surface they are on is ripped or damaged, impairing their visibility. They also need a fair amount of lighting to be read efficiently and cannot be modified to fit the aesthetic of the surface they are on [3].

NFC tags are much easier to interact with, requiring just a mobile device tap with no visual restrictions, and are more durable. The plastic outer layer of the tags serves only as a protective barrier and can even be modified. This ability for modification allows them to be painted or designed to fit any intended business image. Another issue of note is the increased security of NFC technology. QR Codes can easily be cloned with a simple picture of the tag, whereas cloning an NFC tag is much more difficult. The technology's different locking mechanisms and password protection lead to a more secure process.

Looking at future work for this application and the analysis of this technology, the intended goal is to expand the functionality of the app to support a more robust set of features and examine the possibility of increasing the security capabilities of the NFC technology. As of now, the app developed can only perform a handful of basic written commands. The intended functionality would be for users to be able to write all variations of written commands, adjust lock and password bytes for security purposes, and be able to access a full scan read of all 135 pages of memory organization to ensure every page is configured as needed. The app should be an all-in-one reader and writer with the ability for feature configuration by authorized developers. With this app as a singular foundation, the technology can be built upon to push the boundaries of what it is capable of.

[1] NXP Semiconductors (2015, June.). *NTAG213/215/216 NFC Forum Type 2 Tag Complaint IC with 144/504/888 Bytes User Memory* (Rev. 3.2) [Online] Available: https://www.nxp.com/docs/en/data-sheet/NTAG213_215_216.pdf [Accessed April 3rd,2023]

[2] MAKA RFID, "5 Forum Types of NFC Tag", *NFC Guide: In-Depth Read About Near-Field Communication (NFC),* 2019

[3] Ursula Schilling, "*NFC Money Transfer Specification Vs. QR Codes Focus of Digital Payments South Asia Presentation*", in *Digital Payments South Asia 2019,* Mumbai, India, August 29-30, 2019