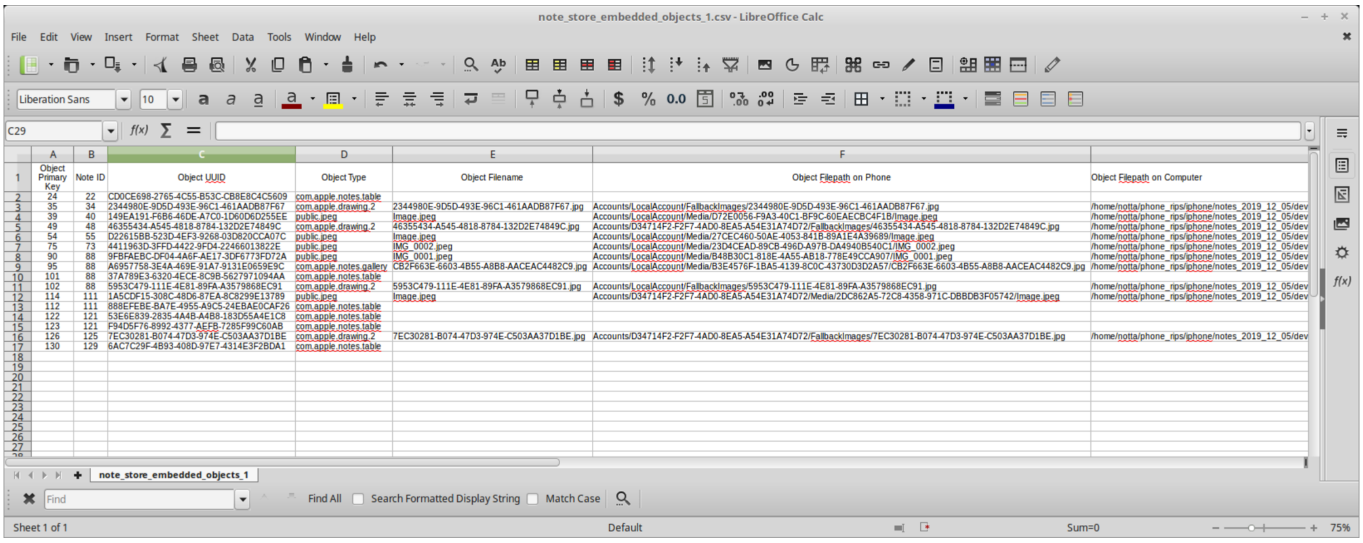
CIOFECA FORENSICS

SOLUTIONS MOBILE APPLE NOTES

Revisiting Apple Notes (1): Improved Note Parsing

10 Jan 2020 · 11 mins read


note_store_embedded_objects_1.csv - LibreOffice Calc
File Edit View Insert Format Sheet Data Tools Window Help
Liberation Sans 10
C29
note_store_embedded_objects_1
Find All Search Formatted Display String Match Case
Sheet 1 of 1 Default Sum=0 75%

Object Primary Key	Note ID	Object UUID	Object Type	Object Filename	Object Filepath on Phone	Object Filepath on Computer
24	22	CD0CE698-2765-4C55-B53C-CB8E8C4C5609	com.apple.notes.table			
35	34	2344980E-9D5D-493E-96C1-461AADB87F67	com.apple.drawing.2	2344980E-9D5D-493E-96C1-461AADB87F67.jpg	Accounts/LocalAccount/FallbackImages/2344980E-9D5D-493E-96C1-461AADB87F67.jpg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
39	40	149EA191-F6B6-46DE-A7C0-1D60D6D255EE	public.jpeg	Image.jpeg	Accounts/LocalAccount/Media/D72E0056-F9A3-40C1-BF9C-60EAECBC4F1B/Image.jpeg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
49	48	46355434-A545-4818-8784-132D2E74849C	com.apple.drawing.2	46355434-A545-4818-8784-132D2E74849C.jpg	Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/FallbackImages/46355434-A545-4818-8784-132D2E74849C.jpg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
54	55	D22615BB-523D-4EF3-9268-03D820CCA07C	public.jpeg	Image.jpeg	Accounts/LocalAccount/Media/27CEC460-50AE-4053-841B-89A1E4A39689/Image.jpeg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
75	73	4411963D-3FFD-4422-9FD4-22466013822E	public.jpeg	IMG_0002.jpeg	Accounts/LocalAccount/Media/23D4CEAD-89CB-496D-A97B-DA4940B540C1/IMG_0002.jpeg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
90	88	9FBFAEBC-DF04-4A6F-AE17-3DF6773FD72A	public.jpeg	IMG_0001.jpeg	Accounts/LocalAccount/Media/B48B30C1-818E-4A55-AB18-778E49CCA907/IMG_0001.jpeg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
95	88	A6957758-3E4A-469E-91A7-9131E0659E9C	com.apple.notes.gallery	CB2F663E-6603-4B55-A8B8-AACEAC4482C9.jpg	Accounts/LocalAccount/Media/B3E4576F-1BA5-4139-8C0C-43730D3D2A57/CB2F663E-6603-4B55-A8B8-AACEAC4482C9.jpg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
101	88	37A789E3-6320-4ECE-8C9B-5627971094AA	com.apple.notes.table			
102	88	5953C479-111E-4E81-89FA-A3579868EC91	com.apple.drawing.2	5953C479-111E-4E81-89FA-A3579868EC91.jpg	Accounts/LocalAccount/FallbackImages/5953C479-111E-4E81-89FA-A3579868EC91.jpg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
114	111	1A5CDF15-308C-48D6-87EA-8C8299E13789	public.jpeg	Image.jpeg	Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Media/2DC862A5-72C8-4358-971C-DBBDB3F05742/Image.jpeg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
112	111	888EFEBE-BA7E-4955-A9C5-24EBAE0CAF26	com.apple.notes.table			
122	121	53E6E839-2835-4A4B-A4B8-183D55A4E1C8	com.apple.notes.table			
123	121	F94D5F76-8992-4377-AEFB-7285F99C60AB	com.apple.notes.table			
126	125	7EC30281-B074-47D3-974E-C503AA37D1BE	com.apple.drawing.2	7EC30281-B074-47D3-974E-C503AA37D1BE.jpg	Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/FallbackImages/7EC30281-B074-47D3-974E-C503AA37D1BE.jpg	/home/notta/phone_rips/iphone/notes_2019_12_05/dev
130	129	6AC7C29F-4B93-408D-97E7-4314E3F2BDA1	com.apple.notes.table			

TL;DR: Apple iCLoud Notes are GZIP'd protobufs when stored and this updated program will decompress them for you and help you understand how to display them closer to the original, displaying them with original formatting and images.

Background

Two years ago, after going through SANS FOR585, I put out a small Perl script to better parse the "new" version of Apple Notes which gzipped its contents instead of storing them plaintext. One of the requests I have heard for the script since then was to pair the images included in Notes since iOS 9 with the note itself. In reviewing the code and digging into the issue, I learned a lot more about how Apple Notes works, the differences between the versions in function and storage, and came up with a more fully-featured program to handle this better. More of those details will come in later blog posts, this one focuses on the new version of Apple Notes Parser.

Apple Notes Data Formats After iOS 9

previous work would guess as to the location and length of the plaintext in that protobuf, without handling it as a protobuf, because, frankly, I had not recognized it as such until I read this post on Mac4n6. After reading that, it is clear Apple Notes uses protobufs in a few places in their database, most notably to store the note's data and to store embedded object data, but mroe on that in future articles.

Apple Notes Parser Updates

Language

The new version of the program has a lot of changes under the hood and is definitely a breaking change to its predecessor due to the change in languages. As I looked to parse the protobuf I came up against the issue that Perl is not an officially supported language by Google. While there is a Perl module that adds this functionality in (Google::ProtobolBuffers), it hasn't been updated in a few years and didn't work on current examples. Since Ruby is an officially supported language, is a language I enjoy, and would scale better for future additions, I opted to port the code over to that.

While re-writing the parser into a Ruby program, I made it fully object oriented. This makes it much easier to fix and extend for the future (and I'd seriously suggest those who just know how to write a good enough script to get the job done look into some programming theory to see how much time you can save in the long run by designing it right from the start, not that I'll claim this is). This also means that others who want to interact with notes can use the base classes as a starting point to do something different.

For example, the AppleNote class represents a note and holds all the necessary information to interact with that note. AppleNote, AppleNotesAccount, AppleNotesFolder, and AppleNotesEmbeddedObject control the CSV output, each using a function named `to_csv`. AppleNote's is:

```
def to_csv
  [@primary_key,
   @note_id,
   get_account_name,
   get_folder_name,
   @title,
   @creation_time,
   @modify_time,
   @plaintext,
   @is_password_protected,
   @crypto_iterations,
   get_crypto_salt_hex,
   get_crypto_tag_hex,
   get_crypto_key_hex,
```

CIOFECA FORENSICS

If you thought there were too many columns and wanted to get rid of the encryption information in favor of just knowing how many embedded objects there were, you could change it to this (note the change on the last line):

```
def to_csv
  [@primary_key,
   @note_id,
   get_account_name,
   get_folder_name,
   @title,
   @creation_time,
   @modify_time,
   @plaintext,
   @embedded_objects.length]
end
```

Added Functionality

The new version preserves all of the old functionality. You can still run it with no commands to parse a NoteStore.sqlite that is in the same folder. However,it also has some new functionality. If you point it at an iTunes backup folder, it will identify the NoteStore.sqlite (hashed to 4f98687d8ab0d6d1a371110e6b7300f6e465bef2) and parse that file. But it will also identify all the embedded images from the NoteStore.sqlite file that remain in the backup and pull those out as well for examination.

Now when you run the program, the following happens (this may change slightly over time, whatever is in the Github repo is definitive):

1. An **AppleBackup** object is created based on command line arguments (currently either an iTunes backuup folder, or just a NoteStore.sqlite file)
2. The **AppleBackup** object creates an AppleNoteStore object that handles the NoteStore file(s)
3. The **AppleNoteStore** object guesses which iOS version it came from based on the structure of the database it is pointing to.
4. The **AppleNoteStore** object rips the accounts from the sqlite database, creating individual **AppleNotesAccount** objects for each.
5. The **AppleNoteStore** object rips the folders from the sqlite database, creating individual **AppleNotesFolder** objects for each.
6. The **AppleNoteStore** object rips the notes from the sqlite database, creating individual **AppleNote** objects for each.
    1. Each **AppleNote** object pulls information from both ZICCLOUDSYNCINGOBJECT and ZICNOTEDATA (and Z_11NOTES for iOS 11) to track most of the relevant information.

    data.
    4. If successful, it then attempts to parse the protobuf that should be inside.
    5. If successful, it stores the plaintext in another column in the NoteStore's ZICNOTEDATA table (ZPLAINTEXTDATA), and adds the plaintext to the **AppleNote** object.
    6. If successful, it then scans the plaintext and protobuf for embedded objects, and creates AppleNotesEmbeddedObject objects for each that it finds.
7. At the end, the program creates an output directory that contains:
    1. A `csv` folder for four CSV files summarizing the **AppleNotesAccount**, **AppleNotesFolder**, **AppleNote**, and **AppleNotesEmbeddedObject** objects.
    2. A `files` directory if an iTunes backup was used, containing copies of the pictures that were embedded in the notes, following the file path they should have.
    3. An `html` directory that contains an HTML representation of that **AppleNoteStore** (i.e. the Folders and Accounts, with Notes and content formatted as they were originally).
    4. A copy of the NoteStore.sqlite file that was made before all this began, to leave the original intact.
    5. A copy of the notes.sqlite file that was made before all this began, to leave the original intact.
    6. A copy of the Manifest.db if an iTunes backup was used, to leave the original intact.

For embedded objects, the program tries to represent them as faithfully as possible. All will have at least the object type (such as public.jpeg) and the object's UUID, which can be looked up in the ZICNOTEDATA.ZIDENTIFIER column. Pictures will identify where they are on disk and tables will identify the text in each cell. In the HTML output, pictures refer to the thumbnail stored by Notes, although it will also copy out the fullsize image, and tables are rendered as a table.

Requirements and Usage

This program now requires Ruby, instead of Perl, which concerned me at first since I consider Perl to be fairly ubiquitous and Ruby not as much. However, this code doesn't have many dependencies, those it does are generally old and well maintained gems, and the backwards compatibility of Ruby means this code will run on versions far older than I have pinned it. Right now the program is expecting Ruby 2.3.0 or newer (~2015), but for those with an older version that want to decrement the version of Google Protobufs used, it should work back to 2.0 at least. Ruby 2.7 was just released and Google Protobufs is not yet compiled for it, until that occurs, I'd pinned it to require a version less than 2.7. The few gems that are required are all on the official Ruby gems repo and shouldn't have any surprises.

```
    -p, --physical DIRECTORY         Root directory of a physical backup (i.e. right
    -o, --output-dir DIRECTORY       Change the output directory from the default ./
    -h, --help                       Print help information
```

Basic

To use the new version the same as the old, put a NoteStore.sqlite file into the root directory of the program and type `rake` into the command line. Although it doesn't appear in the above, Rake is the Ruby version of the classic Make tool, which is basically what real programmers used before IDEs automated everything and we just docker'd our saltstack using a repo of someone else's code (speaking in jest).

```
notta@cuppa ~/apple_cloud_notes_parser $ rake
/usr/bin/ruby2.3 notes_cloud_ripper.rb --file NoteStore.sqlite

Starting Apple Notes Parser at Fri Jan 10 13:46:36 2020
Storing the results in ./output/2020_01_10-13_46_36

Created a new AppleBackup from single file: NoteStore.sqlite
Guessed Notes Version: 11
Updated AppleNoteStore object with 4 AppleNotes in 3 folders belonging to 1 accounts
Adding the ZICNOTEDATA.ZPLAINTEXT and ZICNOTEDATA.ZDECOMPRESSEDDATA columns, this ta

Successfully finished at Fri Jan 10 13:46:37 2020
```

In this example, `rake` was expanded to run ruby on the `notes_cloud_ripper.rb` file, passing in the `--file NoteStore.sqlite` argument to identify the file you parse. If you'd like to get more particular with the arguments, you'll want to directly invoke ruby and specify the arguments to use.

iTunes / Logical

```
notta@cuppa ~/apple_cloud_notes_parser $ ruby notes_cloud_ripper.rb --itunes-dir ~/p

Starting Apple Notes Parser at Fri Jan 10 13:48:01 2020
Storing the results in ./output/2020_01_10-13_48_01

Created a new AppleBackup from iTunes backup: /home/notta/phone_rips/iphone/notes_20
Guessed Notes Version: 13
Guessed Notes Version: 8
Updated AppleNoteStore object with 43 AppleNotes in 6 folders belonging to 2 account
Updated AppleNoteStore object with 0 AppleNotes in 1 folders belonging to 1 accounts
Adding the ZICNOTEDATA.ZPLAINTEXT and ZICNOTEDATA.ZDECOMPRESSEDDATA columns, this ta
```

CIOFECA FORENSICS

In this example, `ruby notes_cloud_ripper.rb` executed with the argument to find an iTunes backup directory located at `--itunes-dir ~/phone_rips/iphone/notes_2019_12_05/device_id/`. This would be the root of the iTunes backup, with `Manifest.db` present.

Physical

```
notta@cuppa ~/apple_cloud_notes_parser $ ruby notes_cloud_ripper.rb --physical ~/pho

Starting Apple Notes Parser at Fri Jan 10 13:57:53 2020
Storing the results in ./output/2020_01_10-13_57_53

Created a new AppleBackup from physical backup: /home/notta/phone_rips/iphone/iOS13_
Guessed Notes Version: 13
Guessed Notes Version: 8
Updated AppleNoteStore object with 42 AppleNotes in 6 folders belonging to 2 account
Updated AppleNoteStore object with 0 AppleNotes in 0 folders belonging to 0 accounts
Adding the ZICNOTEDATA.ZPLAINTEXT and ZICNOTEDATA.ZDECOMPRESSEDDATA columns, this ta

Successfully finished at Fri Jan 10 13:57:55 2020
```

In this example, `ruby notes_cloud_ripper.rb` executed with the argument to find a physical backup directory located at `~/phone_rips/iphone/iOS13_tar/` (how you obtain that physical backup is up to you. This would be the root of the physical backup, with the `private` directory under it (for the sake of hard drives, the important thing really is to export the /private directory).

Power to the Pictures!

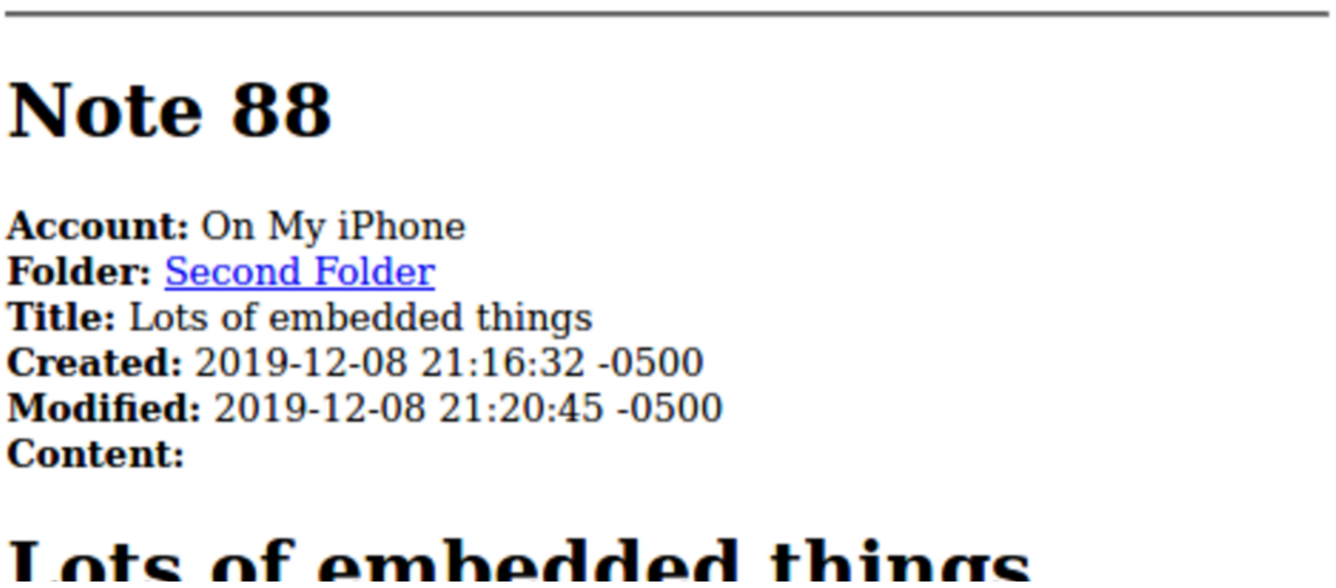
To show the value in running this on real backups, below is an example of the files created from the iTunes example above. Because we used a full backup, the output directory now includes the pictures, thumbnails, and drawings that were embedded in the notes:

```
notta@cuppa ~/apple_cloud_notes_parser/output/2019_12_21-19_21_47 $ find -type f

./NoteStore.sqlite
./notes.sqlite
./html/all_notes_1.html
./html/all_notes_2.html
./files/Accounts/LocalAccount/FallbackImages/2344980E-9D5D-493E-96C1-461AADB87F67.jp
./files/Accounts/LocalAccount/FallbackImages/5953C479-111E-4E81-89FA-A3579868EC91.jp
./files/Accounts/LocalAccount/Media/D72E0056-F9A3-40C1-BF9C-60EAECBC4F1B/Image.jpeg
./files/Accounts/LocalAccount/Media/23D4CEAD-89CB-496D-A97B-DA4940B540C1/IMG_0002.jp
./files/Accounts/LocalAccount/Media/B3E4576F-1BA5-4139-8C0C-43730D3D2A57/CB2F663E-66
```

```
./files/Accounts/LocalAccount/Previews/149EA191-F6B6-46DE-A7C0-1D60D6D255EE-1-384x28
./files/Accounts/LocalAccount/Previews/B6C10486-A0C3-4CFC-8E05-039D0E314AF3-1-192x12
./files/Accounts/LocalAccount/Previews/9FBFAEBC-DF04-4A6F-AE17-3DF6773FD72A-1-192x14
./files/Accounts/LocalAccount/Previews/2344980E-9D5D-493E-96C1-461AADB87F67-1-768x76
./files/Accounts/LocalAccount/Previews/D22615BB-523D-4EF3-9268-03D820CCA07C-1-384x28
./files/Accounts/LocalAccount/Previews/5953C479-111E-4E81-89FA-A3579868EC91-1-768x76
./files/Accounts/LocalAccount/Previews/4411963D-3FFD-4422-9FD4-22466013822E-1-384x28
./files/Accounts/LocalAccount/Previews/B6C10486-A0C3-4CFC-8E05-039D0E314AF3-1-2645x1
./files/Accounts/LocalAccount/Previews/149EA191-F6B6-46DE-A7C0-1D60D6D255EE-1-192x14
./files/Accounts/LocalAccount/Previews/9FBFAEBC-DF04-4A6F-AE17-3DF6773FD72A-1-384x28
./files/Accounts/LocalAccount/Previews/5953C479-111E-4E81-89FA-A3579868EC91-1-768x76
./files/Accounts/LocalAccount/Previews/B6C10486-A0C3-4CFC-8E05-039D0E314AF3-1-384x24
./files/Accounts/LocalAccount/Previews/2344980E-9D5D-493E-96C1-461AADB87F67-1-768x76
./files/Accounts/LocalAccount/Previews/D22615BB-523D-4EF3-9268-03D820CCA07C-1-192x14
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/FallbackImages/7EC30281-B074-4
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/FallbackImages/46355434-A545-4
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Media/2DC862A5-72C8-4358-971C-
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/7EC30281-B074-47D3-97
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/1A5CDF15-308C-48D6-87
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/1A5CDF15-308C-48D6-87
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/46355434-A545-4818-87
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/7EC30281-B074-47D3-97
./files/Accounts/D34714F2-F2F7-4AD0-8EA5-A54E31A74D72/Previews/46355434-A545-4818-87
./csv/note_store_notes_1.csv
./csv/note_store_notes_2.csv
./csv/note_store_accounts_2.csv
./csv/note_store_folders_1.csv
./csv/note_store_accounts_1.csv
./csv/note_store_folders_2.csv
./csv/note_store_embedded_objects_1.csv
./csv/note_store_embedded_objects_2.csv
./Manifest.db
```

And this is a screenshot of one of the notes in that HTML export:



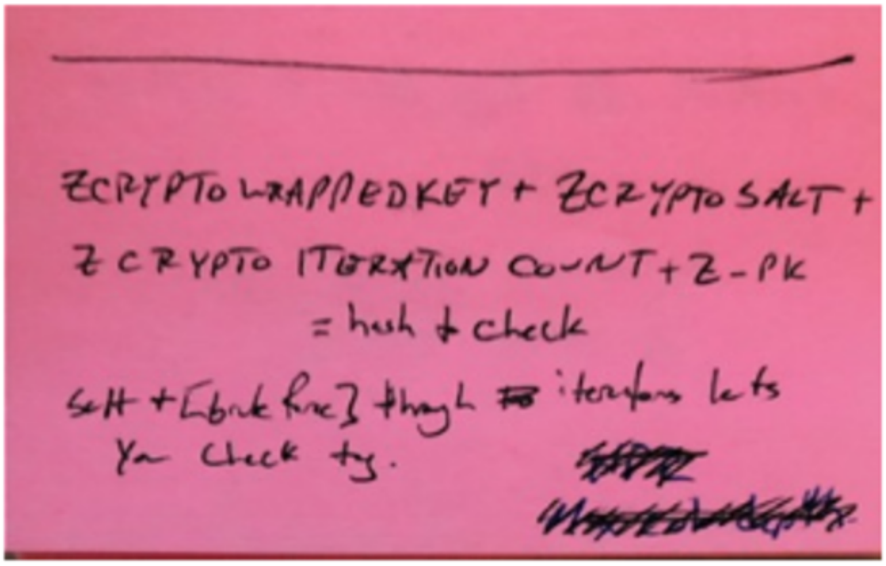
Note 88

Account: On My iPhone
Folder: Second Folder
Title: Lots of embedded things
Created: 2019-12-08 21:16:32 -0500
Modified: 2019-12-08 21:20:45 -0500
Content:

Lots of embedded things

CIOFECA FORENSICS

First pic


ZCRYPTOWRAPPEDKEY + ZCRYPTOSALT +
ZCRYPTO ITERATION COUNT + Z-PK
= hash + check
salt + [brute force] though ~~to~~ iterations lets
you check try.
[redacted]

Scanned document

Table r1c1	R1c2	R1c3
R2c1	R2c2	R2c3

Tables above

1. Ordered list
2. Ordered list 2

✓ Checkbox checked
○ Checkbox unchecked

Checkboxes above

CIOFECA FORENSICS

Conclusion

At the end of the day, I hope this update makes the Notes Parser more useful. While this was a majorly breaking change, it should still have the functionality from the previous version, but with added features to get after embedded objects better. Additional posts will help lay out what the Notes formats and how they've changed over the years.

Previous
‹ Make Analysis Great Again (...

Next
Revisiting Apple Notes (2):... ›