citocrypt User's Guide & Information Manual

Dave Lohmoeller
CITO onDemand, LLC
dave@cito-ondemand.com
PH: (765) 447-7958
Web: cito-ondemand.com

Overview:

citocrypt is a proprietary cryptography software product developed by CITO onDemand, LLC to encrypt data streams. It is designed to be used as part of other programs such as web based scripts. It will take the input from the screens, run it through the program and return the data in an encrypted format. Here is an example:

```
cito-linux.com[  ]lomiller:
 > /home/lomiller/register
 50 cito-example.pl

Encrypting data is taking that which is human readable like:
         dave@cito-ondemand.com
and making it totally unrecognizable like this:
         ?Ces4EJv4gMhh1oxWeqjgpWG68122.PXlN%REq4{47PN9X7PYR49[

So Command:      citocrypt -e cito -s 'dave@cito-ondemand.com'
Produces:        ?Ces4EJv4gMhh1oxWeqjgpWG68122.PXlN%REq4{47PN9X7PYR49[

And to reverse it:
Command:         citocrypt -d cito -s ?Ces4EJv4gMhh1oxWeqjgpWG68122.PXlN%REq4{47PN9X7PYR49['
Produces:        dave@cito-ondemand.com

cito-linux.com[  ]lomiller:
 > /home/lomiller/register
```

In this example, the string “dave@cito-ondemand.com” is ran through the citocrypt tool and returns the encrypted string: ?Ces4EJv4gMhh1oxWeqjgpWG68122.PXlN%REq4{47PN9X7PYR49[

In order to return it to its natural, human readable state, you need to reverse the process using the same key, in this example “cito” and it will decipher and return “dave@cito-ondemand.com”.

You as the programmer have a lot of flexibility in how to use the program. For instance, you can hardcode the key in your web scripts, you can store them in a database and have the variable selected each time, you can designate each user to own their piece of data and supply the key each time they want to make changes, the list can go on. Simply put, the level of security is where it should be, in your hands.

The product was designed with databases in mind since that is where a considerable amount of hacks occur. If you have an existing database, you will need to create a program to run against the database while offline and encrypt the columns you want encrypted. Again, this is part of the flexibility, you choose what to encrypt. Personal Information (PI) data is highly suggested as well as financial components such as social security numbers, credit card numbers, even telephone and cell phone numbers. This works with the flexibility aspect in that an authorized user can supply their own KEY, or be authorized to use a provided key while someone not authorized could still view a partial listing.

Note: When updating an existing database, take the time needed to analyze the column definitions to ensure they are wide enough to handle the expanded output citocrypt provides. A simple Perl, Python, Java or VBA script should handle most of the updates seamlessly. In the testing process, the time expended to generate a core string, encrypt, decrypt and then compare the before and after results were 0.3219 seconds for a 4K string. That was the average over multiple runs of 250K using the maximum string length of 4K. The 1K and 2K tests were only marginally quicker but over the duration of the run, it was noticeable.

Use in a web script:

Since this was designed with databases in mind, portability, speed and accuracy were paramount concerns. I write most of my cgi’s in Perl so it was a simple call to the citocrypt module to perform the encryption. While I haven’t had the

need yet, I could set my forms up so that I had an input field available where the user could enter an encryption key and use it to encrypt the data in the database that they entered. I could also create a user profile with different access levels and assign a different encryption key to each access level so that as they manipulated data the database columns they owned, the key they were intended to use would be used. This would allow multiple levels of encryption within the same rows of database data allowing different employees access to relevant data to perform their jobs. This could simplify many tables and reports where multiple people have to provide input.

Testing:

This program has undergone many hours of testing. The methodology employed was to run a script in Linux that generated a random string of characters, spaces and special characters in increments of 1 to 1k, 1 to 2k and 1 to 4k in length. A static encryption key was used throughout the testing. The following steps then ensued:

1. Variable length string created and stored as a variable (V1)
2. Variable (V1) passed to citocrypt and the encrypted value was returned to the program as a separate variable (V2)
3. Variable V2 was passed back to citocrypt for decryption and the return value stored as V3
4. V3 was compared to V1 and if they matched exactly, then the test was passed.
5. This was repeated in 250K increments for 1K, 2K and 4K.
6. All passed without issue multiple times, well over 1 million iterations without a failure

Anytime the slightest change is made to the program whether or not it involved the encryption/decryption algorithm, this testing process is followed. When the program is made available, at minimum 1 million iterations have taken place without a failure. **This is a critical step for obvious reasons.**

As mentioned earlier, when these are tests are executing, the timing of each step is tracked. Best speed noted so far is 0.3219 seconds for the 4 steps listed above. This was on a lightly loaded, Red Hat Linux server with 3gb memory, 2 cpu's at 2.67ghz running in 64 bit mode.

Limitations:

The only known limitations have to deal with the character set that HTML has set aside for reserved characters. These characters are excluded from use due to the obvious reason that when used, they create issues on Web screens.

The **key** is limited to 19 characters in length using the same character set.

Examples:

Here is an example of data in an actual database. The first four entries are traditional, unencrypted and the last four are encrypted. Both data streams are sourced from identical data.

```
MariaDB [test]> select fname,lname,email from sample_data;
+-------+-----------------------------------------+-----------------------------------------------------------------+
| fname | lname                                   | email                                                           |
+-------+-----------------------------------------+-----------------------------------------------------------------+
| Dave  | Lohmoeller                              | dave@cito-oncemand.com                                          |
| Cyndi | Lohmoeller                              | cyndi@cito-ondemand.com                                         |
| Bob   | Smith                                   | smith@fubar.com                                                 |
| Carol | Johnson                                 | carol.johnson@longhostname.com                                  |
| Dave  | ?Ces8JOA919Kb922HpqGN2kYVz310.56I*6P++Px[ | ?Ces80vhQ3QrTQH2b79FTnvm5J822.W~sU.YLx*b*+YUB~+W|Y*B[           |
| Cyndi | ?Ces8HMy7j7I_7_WgGhbFsAc72410.67J07QAAQy[ | ?Ces87CoX_Xy0XkNAbRSNA0jiQ123.Re7PE%REq4{47PN9X7PYR49[          |
| Bob   | ?Ces8JOA919Kb9o3gk4uPA0Awi75.LAKwMfdM*c[  | ?Ces8CHt2e2D52xxKYI6mL60Gf415.w0HtJ_OrX y@U70[                   |
| Carol | ?Ces8CHt2e2D52wHwJXWeRfEH657.28K*x8*raj[  | ?Ces8DIu3f3E63uHB313W1LcJr130.RXv40YC4G7t47%047JG4tq7X9NYR49[ |
+-------+-----------------------------------------+-----------------------------------------------------------------+
8 rows in set (0.00 sec)

MariaDB [test]> select fname,lname,email from sample_data;
```

In the example, I chose columns lname (last name) and email (email address) to keep from public visibility.

Note: The encrypted lines all begin with "?Ces" which would allow the reports programmer to test for encrypted datastreams and exclude them from viewing.

In this example, I have lname (last name) and addr1 (street address) encrypted.

```
MariaDB [test]> select fname,lname,addr1 from sample_data;
+-------+-----------------------------------------+-----------------------------------------------------+
| fname | lname                                   | addr1                                               |
+-------+-----------------------------------------+-----------------------------------------------------+
| Dave  | Lohmoeller                              | 1237 Meadowbrook Dr                                 |
| Cyndi | Lohmoeller                              | 1237 Meadowbrook Dr                                 |
| Bob   | Smith                                   | 12345 Any Street                                    |
| Carol | Johnson                                 | 987 Bunker Hill Dr                                  |
| Dave  | ?Ces8JOA919Kb922HpqGN2kYVz310.56I*6P++Px[ | ?Ces88DpYaYz1Y8XX7tAMto1q1619.mwN6~-S,U9oZ199E~n1[   |
| Cyndi | ?Ces8HMy7j7I_7_WgGhbFsAc72410.67J07QAAQy[ | ?Ces89EqZbZA2Z7yqqE9knW8m5619.mwN6~-S,U9oZ199E~n1[   |
| Bob   | ?Ces8JOA919Kb9o3gk4uPA0Awi75.LAKwMfdM*c[  | ?Ces8BGs1d1C41tiJ1ODuz6WK0416.kuLqz,e-h,ItyQQt[      |
| Carol | ?Ces8CHt2e2D52wHwJXWeRfEH657.28K*x8*raj[  | ?Ces8GLx6i6H96bmEoopUkPW3f118.6-1Zdo7+NvZsE00Ziv[    |
+-------+-----------------------------------------+-----------------------------------------------------+
8 rows in set (0.00 sec)

MariaDB [test]>
```

For this example, I wrote a simple Perl script to load data from an Excel spreadsheet saved in csv format. The csv data was raw, I encrypted the fields I wanted as it processed each line of data.

Syntax:

The program is simple in its usage. Simply pass the variables to it as follows:

To encrypt:

citocrypt –e <key> -s <string>

or

$enc_string = `citocrypt –e cito_key –s 'I want to encrypt this string my way, today.'`;

Note in the above example, single quotes were used to wrap the input string for the program!

```
17 citocrypt -e cito_key -s 'I want to encrypt this string my way, today.'
?Ces82xjS5LetFpvCVmtjf5hAt144.uZjX7qZq4ZN7Rve2qZqGEtZtqvE7JZ9eZjXe}Zq4PXeY[cito-linux.com[ ]lomiller:
 > /var/www/cgi-bin
 18 citocrypt -d cito_key -s '?Ces82xjS5LetFpvCVmtjf5hAt144.uZjX7qZq4ZN7Rve2qZqGEtZtqvE7JZ9eZjXe}Zq4PXeY['
I want to encrypt this string my way, today.cito-linux.com[ ]lomiller:
 > /var/www/cgi-bin
 19
```

Notice in the above example, the output ran into the Linux command line! This is because by default the return string does not have any CRLF appended to the string. If you want ease of view or cut and paste capabilities, try this:

```
28 citocrypt -S -e cito_key -s 'I want to encrypt this string my way, today.'
?Ces8AFr0cTmBN0rNFkq7MyKSt144.uZjX7qZq4ZN7Rve2qZqGEtZtqvE7JZ9eZjXe}Zq4PXeY[
cito-linux.com[ ]lomiller:
 > /var/www/cgi-bin
 29 citocrypt -S -d cito_key '?Ces8AFr0cTmBN0rNFkq7MyKSt144.uZjX7qZq4ZN7Rve2qZqGEtZtqvE7JZ9eZjXe}Zq4PXeY['
I want to encrypt this string my way, today.
cito-linux.com[ ]lomiller:
 > /var/www/cgi-bin
```

In the second example, CRLF was added to the output by using the **–S** option. Also note, even using the same encryption key in both examples and the same string for encryption, the output was different. This is also by design, the encrypted output will almost always be different but as long as the encryption key is the same, the program will always be able to decrypt the encrypted string!

Programmers Note: The **key** used to lock the string must NOT contain any spaces!

Licensing:

citocrypt is a licensed product from CITO onDemand, LLC. When you download your copy, you have a 30 day evaluation period. At the end of 30 days, it will cease to function. In order to receive a license, call CITO oDemand, LLC at (765) 447-7958. Be prepared to present your specific information with the most attention on the host. Use the "hostname" command to determine this and preset at license time. Once the fee is paid, you will receive an email at the email address you provided good for one year on the host you licensed it to. The license is non-transferable, each host must have their own license. You can run as many databases as you like and use it as a stand-alone tool but each host must be individually licensed.

To install the license, it is best done as the root ID on *IX systems and Administrator on Windows. If not done by these ID's, you will just receive a warning during the license installation.

Note: When installing for a web site, if your web ID does not have access to the common areas where the license tries to install by default, you CAN manually enter the license information in the same directory where the program file is located. Simply Edit a file named ".citolicense.key" and insert the license information contained in the email and save your file. Check permissions on the file so that web ID can access r/o the file otherwise it will not work.

Command Line options:

There are a couple of command line options you might use. Here is a short list of some of the options available:

-h –H	Help
-V	Display Version and exit
-E -e	Encrypt the string passed
-D -d	Decrypt the string passed
-t	Set Temporary License, good for 30 days
-l	Register a PERManent license, replaces TEMP license
-L	Local only install by root or administrator ID
-c	Check License
-s	{string} This is the line of data to be encrypted or decrypted
-S	Formatted for Screen output with carriage return/linefeed