Encrypted Split No Strings Database

Version 5.46 22 July 2021

This is a significantly updated version of an example application written in response to numerous forum questions about methods of preventing data being stolen from Access databases. For example, this thread at **Access World Forums:** **Prevent Importing ODBC tables from ACCDE**

It is intended to show how the data in an Access database can be made reasonably secure against hackers whilst still allowing full functionality to authorised users.

To demonstrate its use, I am setting a **simple security challenge: export the backend data** to an **external application** such as **Excel.**

The attached example application should behave exactly as any **split database** ...
... BUT there are **no linked tables** and therefore **no connection strings** visible in the **navigation pane** or **MSysObjects** system table etc.
In addition, the data is protected using a **RC4 encryption cipher**.

The application consists of an **ACCDE** frontend (FE) and an **ACCDB** backend (BE) with about 2000 records. For the purposes of this example **both FE and BE must be saved in the same folder**.
Make sure it is a **trusted location**.

The **ACCDE FE** has been locked down with the Access **application interface hidden**, the **navigation pane and ribbon removed** and is encrypted with a simple password **isladogs**. The **ACCDB BE** file is also encrypted with a **different and much stronger password** – not supplied nor required to use the application.

The **backend** datafile contains 1 **'deep hidden' table** (though it would work equally well with a standard table)

The **frontend** database contains several forms and a report. **None** of the forms or the report have **a saved record source**. As the **forms/reports** have no record source, there is no link to the **BE table**. Therefore, there is no **connection string** accessible to that table (other than in the code).

Instead, **disconnected ADO recordsets** (AKA in-memory recordsets) are **created using code** when the object is loaded and **destroyed** when it is closed.
This approach works perfectly for **forms** but cannot be used with **reports**. For that reason, the **report data** is actually a **form** with a **disconnected ADO recordset** used as a **subreport**.

All the fields in the **BE** table except the autonumber **PersonID**primary key field have been **encrypted** using **RC4 encryption** (the **cipher key(s)** have not been supplied ... nor are they needed to use this).

The application also contains several additional security measures to deter hacking using automation. For example:
a) It cannot be opened from a non-trusted location
b) It cannot be run using automation from an external application. Any attempts to do so will cause the application to close after a warning message.
c) Right click context menus have been disabled as have various keyboard shortcuts such as Ctrl-C, Ctrl-V, F12 etc.

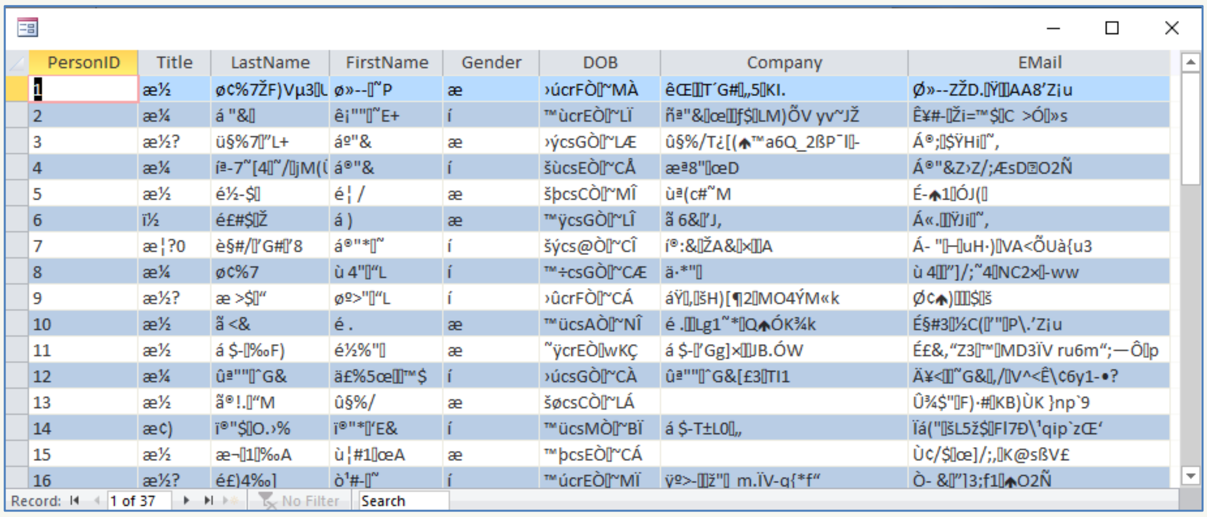
Taken together, **these measures should prevent** anyone being able to **export the data** to an **external application** such as **Excel** (hence the challenge!).

NOTE:
It is **IMPOSSIBLE** to **prevent** anyone taking a **screenshot** of the data. For that reason, the forms/reports should not allow users to view the entire dataset if your data is so sensitive that this approach is used. For that reason, the use of **screenshots** is **NOT** considered an **acceptable solution** to the **security challenge!**

Further Information

1. In this example, the **entire application interface** has been **hidden** so all **forms** are shown **'floating on the desktop'**. The **report** is shown with just the Access title bar - **no ribbon or quick Access toolbar** **(QAT)**. However, doing each of these certainly isn't essential to the idea behind the application

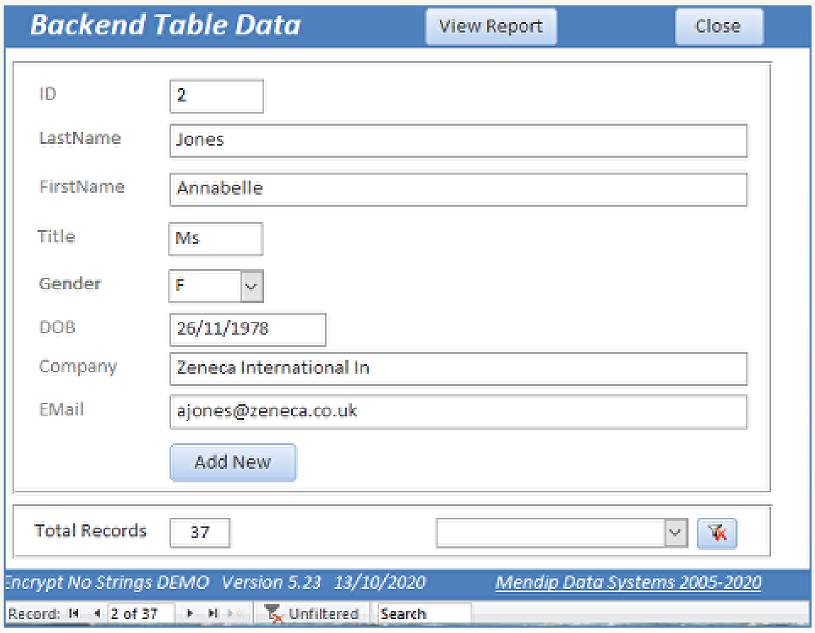
2. I have deliberately left the **encrypted data table** so it can be **viewed** but **NOT directly edited**. As it contains **encrypted data**, editing those fields would lead to **partly encrypted data** being visible in the **form.**



PersonID	Title	LastName	FirstName	Gender	DOB	Company	EMail
1	æ½	ø¢%7ŽF)Vµ3▯L	ø»--▯˜P	æ	›úcrFÒ▯˜MÀ	êŒ▯▯T´G#▯„5▯KI.	Ø»--ZŽD.▯Ÿ▯▯AA8'Z¡u
2	æ¼	á "&▯	ê¡""▯˜E+	í	™ùcrEÒ▯˜LÏ	ñª"&▯œ▯▯ƒ$▯LM)ÕV yv~JŽ	Ê¥#-▯Ži=™$▯C >Ó▯»s
3	æ½?	ü§%7▯"L+	áº"&	æ	›ýcsGÒ▯˜LÆ	û§%/T¿[(⬆™a6Q_2ßP¯I▯-	Á®;▯$ŸHi▯˜,
4	æ¼	íª-7˜[4▯˜/▯jM(Ú	á®"&	í	šùcsEÒ▯˜CÅ	æª8"▯œD	Á®"&Z›Z/;ÆsD▯O2Ñ
5	æ½	é½-$▯	é¦ /	æ	šþcsCÒ▯˜MÎ	ùª(c#˜M	É-⬆1▯ÓJ(▯
6	ï½	é£#$▯Ž	á)	æ	™ÿcsGÒ▯˜LÎ	ã 6&▯'J,	Á«.▯▯ŸJi▯˜,
7	æ¦?0	è§#/▯'G#▯'8	á®"*▯˜	í	šýcs@Ò▯˜CÎ	í®:&▯ŽA&▯×▯▯A	Á- "▯–▯uH·)▯VA<ÕUà{u3
8	æ¼	ø¢%7	ù 4"▯"L	í	™÷csGÒ▯˜CÆ	ä·*"▯	ù 4▯▯"]/;˜4▯NC2×▯-ww
9	æ½?	æ >$▯"	øº>"▯"L	í	›ûcrFÒ▯˜CÁ	áŸ▯,▯šH)[¶2▯MO4ÝM«k	Ø¢⬆)▯▯▯$▯š
10	æ½	ã <&	é .	æ	™ücsAÒ▯˜NÎ	é .▯▯Lg1˜*▯Q⬆ÓK¾k	É§#3▯½C(▯'"▯P\.'Z¡u
11	æ½	á $-▯‰F)	é½%"▯	æ	˜ÿcrEÒ▯wKÇ	á $-▯'Gg]×▯▯JB.ÓW	É£&,"Z3▯™▯MD3ÏV ru6m";—Ô▯p
12	æ¼	ûª""▯ˆG&	ä£%5œ▯▯™$	í	›úcsGÒ▯˜CÀ	ûª""▯ˆG&[£3▯TI1	Ä¥<▯▯˜G&▯,/▯V^<Ê\¢6y1-•?
13	æ½	ã®!.▯"M	û§%/	æ	šøcsCÒ▯˜LÁ		Û¾$"▯F)·#▯KB)ÙK }np`9
14	æ¢)	ï®"$▯O.›%	ï®"*▯'E&	í	™ücsMÒ▯˜BÏ	á $-T±L0▯„	Ïá("▯šL5ž$▯FI7Đ\'qip`zŒ'
15	æ½	æ¬▯1▯‰A	ù¦#1▯œA	æ	™þcsEÒ▯˜CÁ		Ù¢/$▯œ]/;,▯K@sßV£
16	æ½?	é£)4‰]	ò¹#-▯˜	í	™úcrEÒ▯˜MÏ	ÿº>-▯▯ž"▯ m.ÏV-q{*f"	Ò- &▯"]3;f1▯⬆O2Ñ

Record: 1 of 37 No Filter Search

3. Although the **table** contains **encrypted data**, it is displayed **unencrypted** in the **main form** and the **report** The **main form** is **fully editable** and **new records can be added**. Any **changes** will **automatically be encrypted.**

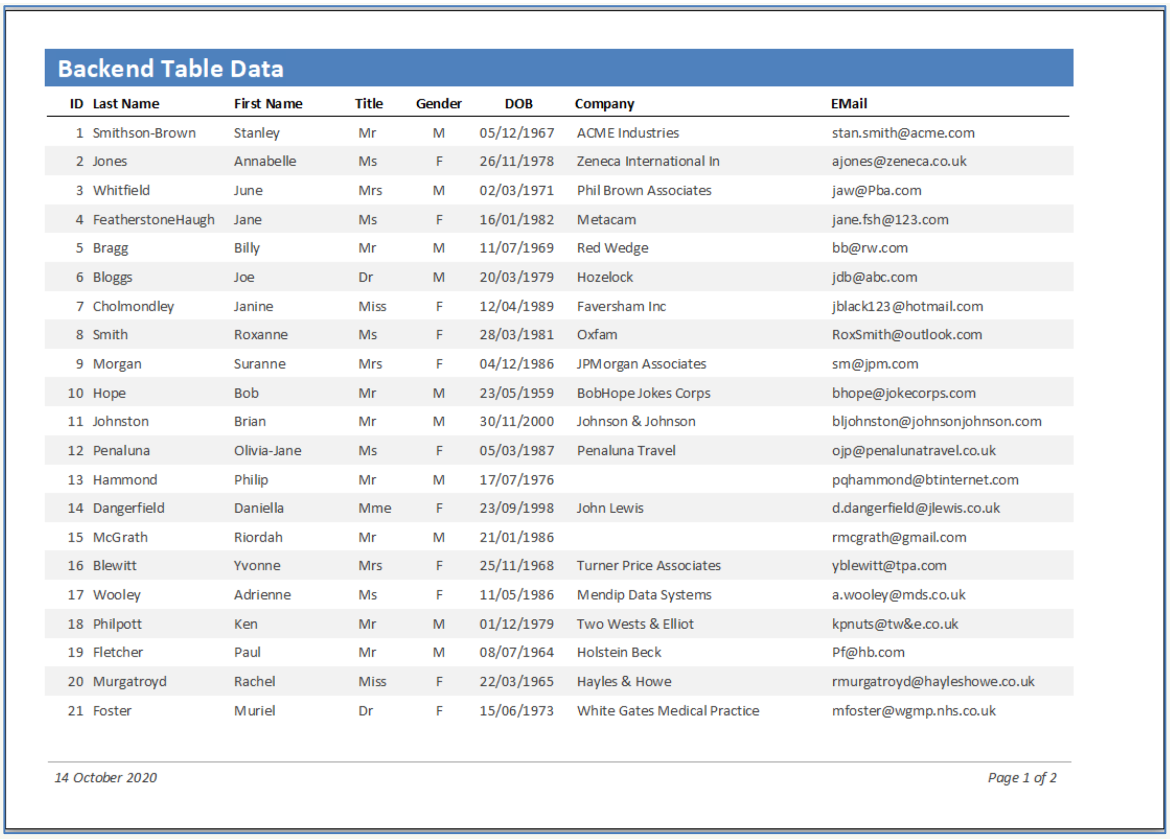


Backend Table Data

ID	2
LastName	Jones
FirstName	Annabelle
Title	Ms
Gender	F
DOB	26/11/1978
Company	Zeneca International In
EMail	ajones@zeneca.co.uk
Total Records	37

Encrypt No Strings DEMO Version 5.23 13/10/2020 *Mendip Data Systems 2005-2020*

Record: 2 of 37 Unfiltered Search



Backend Table Data

ID	Last Name	First Name	Title	Gender	DOB	Company	EMail
1	Smithson-Brown	Stanley	Mr	M	05/12/1967	ACME Industries	stan.smith@acme.com
2	Jones	Annabelle	Ms	F	26/11/1978	Zeneca International In	ajones@zeneca.co.uk
3	Whitfield	June	Mrs	M	02/03/1971	Phil Brown Associates	jaw@Pba.com
4	FeatherstoneHaugh	Jane	Ms	F	16/01/1982	Metacam	jane.fsh@123.com
5	Bragg	Billy	Mr	M	11/07/1969	Red Wedge	bb@rw.com
6	Bloggs	Joe	Dr	M	20/03/1979	Hozelock	jdb@abc.com
7	Cholmondley	Janine	Miss	F	12/04/1989	Faversham Inc	jblack123@hotmail.com
8	Smith	Roxanne	Ms	F	28/03/1981	Oxfam	RoxSmith@outlook.com
9	Morgan	Suranne	Mrs	F	04/12/1986	JPMorgan Associates	sm@jpm.com
10	Hope	Bob	Mr	M	23/05/1959	BobHope Jokes Corps	bhope@jokecorps.com
11	Johnston	Brian	Mr	M	30/11/2000	Johnson & Johnson	bljohnston@johnsonjohnson.com
12	Penaluna	Olivia-Jane	Ms	F	05/03/1987	Penaluna Travel	ojp@penalunatravel.co.uk
13	Hammond	Philip	Mr	M	17/07/1976		pqhammond@btinternet.com
14	Dangerfield	Daniella	Mme	F	23/09/1998	John Lewis	d.dangerfield@jlewis.co.uk
15	McGrath	Riordah	Mr	M	21/01/1986		rmcgrath@gmail.com
16	Blewitt	Yvonne	Mrs	F	25/11/1968	Turner Price Associates	yblewitt@tpa.com
17	Wooley	Adrienne	Ms	F	11/05/1986	Mendip Data Systems	a.wooley@mds.co.uk
18	Philpott	Ken	Mr	M	01/12/1979	Two Wests & Elliot	kpnuts@tw&e.co.uk
19	Fletcher	Paul	Mr	M	08/07/1964	Holstein Beck	Pf@hb.com
20	Murgatroyd	Rachel	Miss	F	22/03/1965	Hayles & Howe	rmurgatroyd@hayleshowe.co.uk
21	Foster	Muriel	Dr	F	15/06/1973	White Gates Medical Practice	mfoster@wgmp.nhs.co.uk

14 October 2020 *Page 1 of 2*

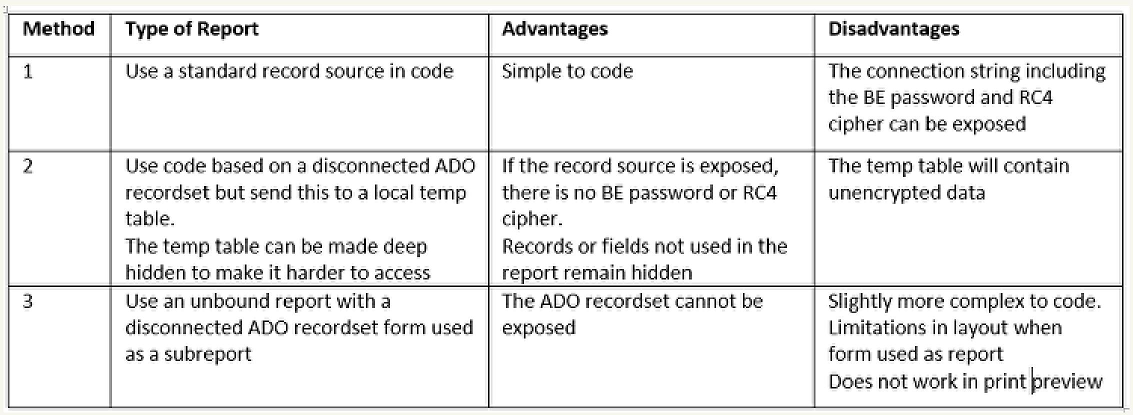
4. Be aware that **creating the editable form** will take **much longer** than usual as **unbound controls** must be used. The form contains **2 sets of each control** in order to **allow editing** of the **encrypted data.** **Unbound controls** are used to display the **decrypted data.** Doing this also means **code** needs to be added to each **unbound control** to **encrypt** the **entered data**

5. If anyone does manage to directly access the **data tables**, all they will see is **encrypted data**. However, it is of course still possible for anyone with **authorised** access to the **FE** to print the decrypted data using a report (if allowed) or just take **screenshots** of the data.

6. This approach is only worth considering if your **data** is **highly sensitive**. If you do want to use this approach with **highly sensitive data** of your own, ensure that:
   a) The **BE** database is given a **different and very strong password** to the FE (see below for password security info
   b) Both the **ribbon and navigation pane are removed** from the FE. All interaction via forms ONLY
   c) End users should **NEVER** be told the BE password
   d) **All data** is stored in the **BE**. There should be **no data tables** in the **FE**
   e) The **FE** is distributed as an **ACCDE** so the code is not accessible
   f) The **Access BE** file is stored **securely on the server** to which **end users** have **no access.** Much better still - use **SQL Server** or similar for the **BE database** as that is several orders of magnitude more secure if properly configured
   g) A **strong 128-bit encryption** method is used such as **RC4** or any **other secure cipher** **XOR** encoding is **NOT recommended** as it is too easy to decode You can use different encryption keys for each table and/or use a different cipher key for each field if it seems worth the additional coding effort needed
   h) **OPTIONAL** - for additional security, the **BE** tables can be **'deep hidden'** as in this DEMO. However, it isn't essential to do so provided you ensure users have no means of accessing the BE

7. You need to be aware that **encrypting date or number fields** is **problematic** as the **encryption cipher converts data** into **'random' text strings**. Hence, I have used a **text field** for the **date of birth (DOB)** field above There is little point encrypting fields with limited values such as Gender (M/F) or Title (Mr/Mrs/Ms/Miss etc). However, I have done so here for completeness.

8. As mentioned earlier, **disconnected ADO recordsets** work well for **forms** but **CANNOT** be used with **reports**. There are various alternative solutions including:



Method	Type of Report	Advantages	Disadvantages
1	Use a standard record source in code	Simple to code	The connection string including the BE password and RC4 cipher can be exposed
2	Use code based on a disconnected ADO recordset but send this to a local temp table. The temp table can be made deep hidden to make it harder to access	If the record source is exposed, there is no BE password or RC4 cipher. Records or fields not used in the report remain hidden	The temp table will contain unencrypted data
3	Use an unbound report with a disconnected ADO recordset form used as a subreport	The ADO recordset cannot be exposed	Slightly more complex to code. Limitations in layout when form used as report Does not work in print preview

Method 3 was used in this version of the example application.

9. **Encryption** and **decryption** are done using the same **RC4 cipher** and **key**. In other words, the **encryption is reversible**. This feature makes coding much easier to manage but it is also a weakness of RC4.
However, **unless the encryption key is known**, it is **almost impossible** to **decrypt the data**

Other ciphers such as **AES** exist where the **encryption** is **'one-way'**, but coding will therefore be **more complex** still.

RC4 Encryption Code

The **RC4 encryption** code used in this application is as follows. It should be placed in a **standard module** e.g. **modRC4Encryption**

```
'#################################################################
'# RC4 encryption function
'# Author: Andreas J”nsson http://www.freevbcode.com/ShowCode.asp?ID=4398
'# RC4 is a stream cipher designed by Rivest for RSA Security.
'#
'# Amended by Colin Riddington / Chris Arnold 14/03/2019 to fix error 9
'#################################################################

Public Function RC4(ByVal Expression As String, ByVal Password As String) As String
  On Error Resume Next

  Dim rb(0 To 255) As Integer, x As Long, Y As Long, z As Long, Key() As Byte, ByteArray() As Byte, temp As Byte

  If Len(Password) = 0 Then
     Exit Function
  End If

  If Len(Expression) = 0 Then
     Exit Function
  End If

  If Len(Password) > 256 Then
     Key() = StrConv(Left$(Password, 256), vbFromUnicode)
  Else
     Key() = StrConv(Password, vbFromUnicode)
  End If
```

```
For x = 0 To 255
    rb(x) = x
  Next x

  x = 0
  Y = 0
  z = 0

  For x = 0 To 255
     Y = (Y + rb(x) + Key(x Mod Len(Password))) Mod 256
     temp = rb(x)
     rb(x) = rb(Y)
     rb(Y) = temp
  Next x

  x = 0
  Y = 0
  z = 0
  ByteArray() = StrConv(Expression, vbFromUnicode)

  'Next line changed after discussion with Chris Arnold as it causes error 9 - subscript out of range
  'Arrays start at 0 so this needs to end with Len(Expression)-1
  ' For x = 0 To Len(Expression)
  For x = 0 To Len(Expression) - 1    'Colin Riddington - 14/03/2019
     Y = (Y + 1) Mod 256
     z = (z + rb(Y)) Mod 256
     temp = rb(Y)
     rb(Y) = rb(z)
     rb(z) = temp
     ByteArray(x) = ByteArray(x) Xor (rb((rb(Y) + rb(z)) Mod 256))
  Next x

  RC4 = StrConv(ByteArray, vbUnicode)

End Function
```

RC4 Encryption examples

Using the same string (gothic19) with 5 different keys:

Org: gothic19	Key: abc	Enc: ªòA1¼ÜJ
Org: gothic19	Key: conundrum	Enc: oÞÈN\jBú
Org: gothic19	Key: a53frt23	Enc: í9 ¿tRõ
Org: gothic19	Key: nempnett67thrubwell	Enc: ˆ´¡¦!“
Org: gothic19	Key: abracadabra	Enc: cïöTcn‰¯

Effect of altering the original string:

Org: Xgothic19	Key: abracadabra	Enc: \çíHbdÛ§°
Org: Xgothic819	Key: abracadabra	Enc: \çíHbdÛ®¸w

Finally, I will repeat some comments I have written many times previously in relation to **security in Access**:

a) Access databases can NEVER be made 100% secure

b) A capable and determined hacker can break any Access database given sufficient time and motivation.

c) However, by erecting various barriers, it is certainly possible to make the process so difficult and time consuming that it isn't normally worth attempting.

d) Access apps (or any applications) are only as secure as the weakest part of the security used

I hope this idea will be interesting for others to use / adapt / improve.

I have used both **encryption** and the **'no strings'** approach for particularly **sensitive data** in both **Access** and **SQL Server** but have never felt it necessary to do so for a whole database. I'll leave others to decide how practical this would be for an entire application!

I would be grateful for any feedback on this article including details of any errors or omissions.
To provide feedback on this application, please contact me by email or use the **feedback form** at the end of this article.

Downloads

a) The **example application** used in this article (Approx 1.2 MB - zipped)

EncryptNoStrings32_365	32-bit ACCDE frontend FEX32.accde & BEX.accdb (Access 365 format)
EncryptNoStrings64_365	64-bit ACCDE frontend FEX64.accde & BEX.accdb (Access 365 format)

b) Encrypted Split No Strings Database This article as a PDF file (approx 0.3 MB - zipped)

Acknowledgements
I would like to thank various Access developers who have suggested ideas and improvements for this application.
Particular thanks are due to:

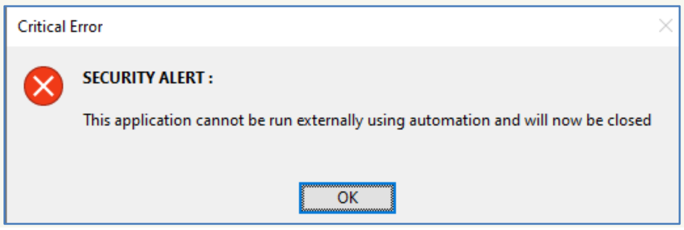
Chris Arnold - for providing valuable assistance with code used with disconnected ADO recordsets

Leo (theDBGuy) - for testing several earlier versions of this application

Philipp Stiefel - for informing me of weaknesses in earlier versions which he used (with my permission) in presentations on Access security to Access developers. One of these presentations can be found on **YouTube** at How (In)Secure is Access Today?- Live at Virtual Access Cascade Conference 2020.
The section starts at around 16 minutes into the presentation. Philipp explains in detail how he was able to use automation to circumvent the security in the earlier version.

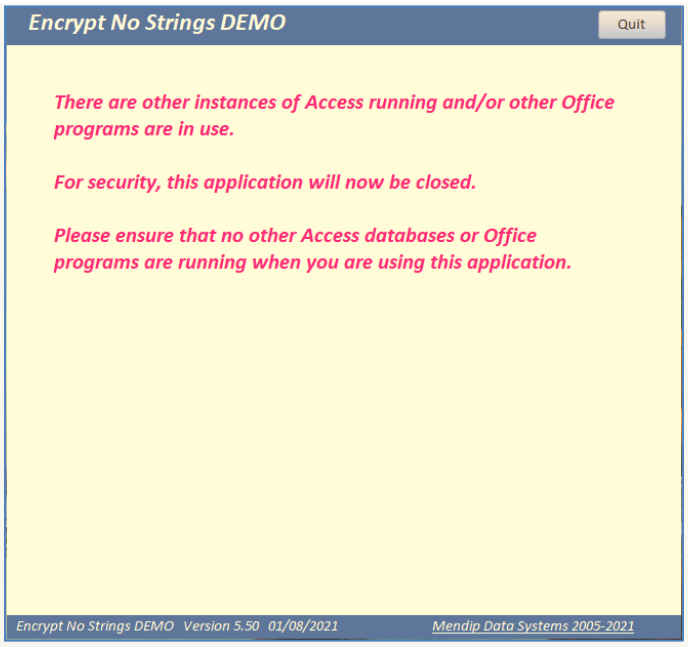
The tests done by Philipp provided the incentive for me to further improve the security in this application!
Since then many further changes have been made. As a result, the methods used by Philipp **no longer work** in the **latest version.**

Depending on how users attempt to gain access from an external program, one of the following messages will be seen and the application will then close automatically.


Critical Error

SECURITY ALERT :

This application cannot be run externally using automation and will now be closed



Encrypt No Strings DEMO

Quit

There are other instances of Access running and/or other Office programs are in use.

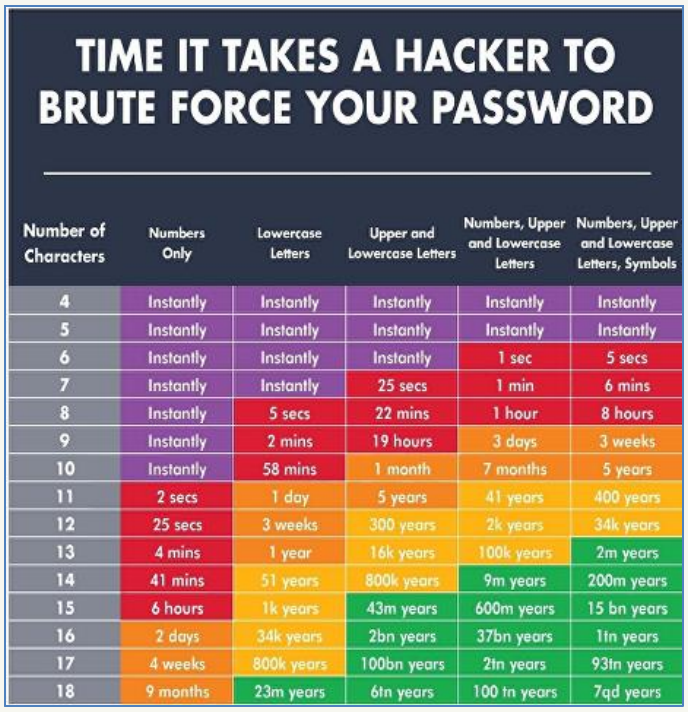
For security, this application will now be closed.

Please ensure that no other Access databases or Office programs are running when you are using this application.

Encrypt No Strings DEMO   Version 5.50   01/08/2021   *Mendip Data Systems 2005-2021*

Password Encryption

ACCDB/ACCDE passwords also **encrypt the entire file** using up to **128-bit encryption.**
For that reason, these passwords can only be broken by a **brute force attack.**
If the password is **reasonably strong** this can be a very lengthy process.



TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years

Chart taken from www.linkedin.com/posts/cbtech-support_ever-wonder-how-long-it-would-theoretically-activity-6715288518240845824-Aiyx

According to https://www.security.org/how-secure-is-my-password the **strong BE password** used in this application would take about **41 trillion years** to crack using a **brute force attack**....in fact slighlty less as I've given you some hints to get you started!!

By comparison, the supplied **FE password (isladogs)** is **very weak** and would only take a **few seconds** to crack. In a real-life application, the FE password would need to be strengthened using a **longer password** containing a **mixture of numbers, upper and lower case letters** and possibly **special characters.**

Further Reading

This article is a companion to the following items on my website:

Access File Security

Improve Security

Purpose of System Tables

Security Challenges

Colin Riddington ***Last updated: 21 Jan 2022***