

Off The Grid

Brute-force Cracking of the
Off The Grid Encryption

Karel Kubat / karel@kubat.nl / www.kubat.nl

What is Off The Grid

- Invented by Steve Gibson / www.grc.com
- Read: www.grc.com/offthegrid.htm
- Read: www.kubat.nl/pages/otg

- Purpose: Encryption that you can perform by hand, but still so strong that computers can't crack it

How it Works

- You create a secret key
- You use the secret key to encrypt or decrypt messages
- OTG is symmetric encryption - the only vulnerability should be the key secrecy; not the algorithm specs or visibility of the encrypted message

OTG's Secret Key: Latin Square

- Latin square: holds all characters that we can encrypt. Characters occur only once per row and column
- Trivial grid:

```
a b c d e f  
b c d e f a  
c d e f a b  
d e f a b c  
e f a b c d  
f a b c d e
```

What is the Grid For?

- OTG encryption "walks" over the grid
- You locate the character to encrypt (input character)
- Every input character produces 2 output characters
- Original purpose: Grid of a-z, used to generate a site password
- E.g., "amazon" -> "VQllefIJCndqc"
- Print out the grid & keep it in your wallet!

Tiny Squares

Trivial

a	b	c	d	e	f
b	c	d	e	f	a
c	d	e	f	a	b
d	e	f	a	b	c
e	f	a	b	c	d
f	a	b	c	d	e

Non-trivial

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

Mixed Case

- Characters are upper or lower case at random
- Without mixed case:
"amazon" -> "vqlfljcnndqc"
- With mixed case:
"amazon" -> "VQlflJcndqc"
- Purpose: more entropy in output, harder to brute-force

How to Generate Your Own Grid

- Create trivial grid (every row shifts by one character)
- Randomly change to upper case
- N times:
 - Swap 2 random rows
 - Swap 2 random columns

How It Works: Encrypting "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at top left
- Find "a"
horizontally
- Output next two characters (eD)
- Move cursor
beyond output (f)
- So far: eD

How It Works: "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at "f"
- Find "b" vertically
- Output next two characters (cd)
- Move cursor beyond output (a)
- So far: eDcd

How It Works: "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at "a"
- Find "c"
horizontally - left!
- Output next two characters (Eb - we scan left)
- Move cursor beyond output (a)
- So far: eDcdEb

How It Works: "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at "a"
- Find "d" vertically - up!
- Output next two characters (cb - we scan up)
- Move cursor beyond output (E)
- So far: eDcdEbc

How It Works: "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at "E"
- Find "e"
horizontally - on it!
- Left or right?
Assume right.
- Output next two characters (fA)
- Move cursor
beyond output (D)
- So far: eDcdEbcbfA

How It Works: "abcdef"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at "D"
- Find "f" vertically
- Left or right?
Assume right.
- Output next two characters (ca)
- Move cursor
beyond output (D)
- So far:
eDcdEbcbfAca

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Take input tuple-wise
- eD cd eb db fA ca
- Every tuple will give us one character

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- Start at topmost row
- Find eD
- Occurs left to right, so:
- Character is "a"
- Cursor at "f"
- So far: "a"

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- In rightmost column
- Find cd
- Occurs downward, so:
- Character is "b"
- Cursor at "a"
- So far: "ab"

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- In bottom row
- Find Eb
- Occurs right to left, so:
- Character is "c"
- Cursor at "a"
- So far: "abc"

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- In rightmost column
- Find cb
- Occurs upward, so:
- Character is "d"
- Cursor at "E"
- So far: "abcd"

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- In second row
- Find fA
- Occurs left to right, so:
- Character is "E"
- Cursor at "D"
- So far: "abcdE"

How to Decrypt "eDcdEbcbfAca"

c	b	a	e	D	f
f	A	D	c	B	E
d	F	e	a	c	b
E	d	b	F	a	c
A	c	F	b	E	d
b	E	c	d	f	a

- In third column
- Find ca
- Occurs downward, so:
- Character is "F"
- Cursor at "D"
- So far: "abcdEF"

How to Attack OTG

- The algorithm is known
- The encrypted message eDcdEbcbfAca is known
- Can we generate a grid that has the tuples?
- Might be the right one...

- Only meaningful with long messages or many short ones!

How to Attack the Grid

.
.
.
.
.
.

- eD cd Eb cb fA ca
- Characters a-f, hence 6x6 grid
- eD must occur left to right in the first row
- cD must occur in the column behind that
- Etc...

How to Attack the Grid

e	D
.
.
.
.
.

- eD cd Eb cb fA ca
- eD must occur left to right in the first row
- E.g., start at top left corner

How to Attack the Grid

e	D	c	.	.	.
c	.	d	.	.	.
a	.	E	b	c	.
.	.	.	.	b	.
.	.	.	.	f	A
.

- eD cd Eb cb fA ca
- eD must occur left to right in the first row
- cd must occur in column 3
- Eb must occur in row 3
- cb must occur in column 5
- fA must occur in row 5
- ca must occur in column 1

How to Attack the Grid

.	A	f	.	e	D
.	.	.	.	b	.
.	.	.	.	c	.
c
d
.	.	E	b	.	.

- eD cd Eb cb fA ca
 - eD must occur left to right in the first row
 - cd must occur in column 1
 - Eb must occur in row 6
 - cb must occur in column 5
 - fA must occur in row 1
 - ca must occur in column 1
- No solution

How to Attack the Grid

.	A	f	.	e	D
.	.	.	.	b	.
.	.	.	.	c	.
c
d
.	.	E	b	.	.

- **ca** must occur in column 1
No solution, but why?
- After **fA** in row 1 scanning can be only downward
- If it were also upward, we could get a/c/d, the c mixed for cd (downward) and ca (upward)
- So the c would be used twice, in two tuples

The Model: Components

- Tiles represent single slots on the grid
- The grid is the y times x board
- The solver does the brute-force attack
- The tuple list is the list of 2-char sequences of encrypted text

The Model: Tile

- Data: value (char), count (unsigned)
- Methods:
 - bool allowed(char c) const - can the tile be set to c? Yes if empty or if value equals c
 - bool conflict(char c) const - is the tile in a neighbor conflict with another tile holding c? Yes if not empty and uppercases don't match
 - void place(char c) - set value to c and increment count if allowed
 - void remove(char c) - decrement count / make empty

The Model: Grid

- Data:
 - 2-dimensional array of tiles
- Methods:
 - `bool allowed(char c1, int y1, int x1, char c2, int y2, int x2)` - is tuple `c1/c2` allowed on tiles `y1/x1` and `y2/x2`?
 - `void place(char c1, int y1, int x1, char c2, int y2, int x2)` - place a tuple
 - `void remove(char c1, int y1, int x1, char c2, int y2, int x2)` - remove a tuple from tiles `y1/x1` and `y2/x2`

The Model: Solver

- **Data:**
 - A grid
 - A tuple list
- **Methods:**
 - `bool solve()`
 - `private bool place(...)`

The Solver: bool solve()

```
bool solve() {  
    return  
        place(0, 0,           // start y/x  
              0,             // tuple index  
              horizontally); // direction  
}
```

The Solver: `bool place(...)`

- Parameters:
 - y/x coordinates to start
 - Index of the tuple to place (0..max)
 - Direction of placement

The Solver: bool place(...)

- Approach:
 - If index is max, return true
 - Otherwise, try to:
 - Place our tuple (see next slide)
 - Call place(...) with index + 1, "next" coordinates, "the other" direction
 - If that succeeds: we're done, return true
 - If that fails: remove our tuple and try next position
 - Return false when possibilities are exhausted

Example: How to Place Horizontally

- When placing given cursor on column x :
 - Try $c1/c2$ to the right
 - $x1$ ranges from x to \max ,
 $x2$ from $x+1$ to $\max+1 (= 0)$
 - Try $c1/c2$ on the leftmost columns
 - $x1$ on 0 , $x2$ on 1 ; wrapped-around right
 - Try $c1/c2$ to the left, in reverse order
 - $x1$ ranges from $x-1$ to 0 ,
 - $x2$ from $x-2$ to $-1 (= \max)$
 - Try $c1/c2$ on the rightmost columns, reverse order
 - $x1$ on \max , $x2$ on $\max-1$; wrapped-around left

How Does This Perform?

- Tests of 5-9 character alphabets
- Random generated grids
- Random generated plaintext input of given alphabet, 5.000 characters
- Encryption of input using the grid
- Cracking attempt using the encrypted text
- Time and result logged

Performance

5-character alphabet a-e

Solution found

Turn: 5497, elapsed: 0.007612 sec

```
  | 0 1 2 3 4
---+-----
0|  c  E  a  b  D
1|  E  B  c  d  A
2|  D  A  b  c  E
3|  A  c  d  E  b
4|  b  d  E  a  C
```

Performance

6-character alphabet a-f

Solution found

Turn: 16040, elapsed: 0.033432 sec

```
| 0 1 2 3 4 5
---+-----
0| f B d a c e
1| d a e c f B
2| B F A D e C
3| C e f b a d
4| A D C e B F
5| E c b f D a
```

Performance

7-character alphabet a-g

Solution found

Turn: 8606, elapsed: 0.017282 sec

```
| 0 1 2 3 4 5 6
-----+-----
0| a b g C D e F
1| B C E G A f d
2| e F a d g b c
3| f D B A E C G
4| c g F e B D a
5| g e D F c A b
6| D A c B f G E
```

Performance

8-character alphabet a-h

Solution found

Turn: 905657, elapsed: 2.91605 sec

```
| 0 1 2 3 4 5 6 7
-----+-----
0| g d C A h b f E
1| f c h e B D A g
2| a H B G d c E F
3| h E g D f a b C
4| E b D f c H G A
5| d F a H e g C b
6| C a E B G f H D
7| b G f c a e d h
```

Performance

9-character alphabet a-i

Solution found

Turn: 1110544016, elapsed: 3714.94 sec

```
| 0 1 2 3 4 5 6 7 8
-----+-----
0| i a c b G h D F e
1| g b E i F D c h A
2| B E d a i f H g c
3| F i a g h c E D B
4| A C H E B g f i d
5| C H g D e B i a F
6| h g B f D E a c I
7| e d F C a I G B h
8| D f i h c a b E g
```

Now what?

- Smarter algorithm
- Parallelize!
- Win the contest

