Handycipher

Bruce Kallick

Curmudgeon Associates
`curmudgeon@rudegnu.com`

1 Introduction

Handycipher is a low-tech, randomized, symmetric-key, stream cipher, simple enough to permit pen-and-paper encrypting and decrypting of messages, while providing a significantly high level of security by combining a simple 31-character substitution cipher with a 3,045-token nondeterministic homophonic substitution cipher. The basic approach of the cipher is to take each plaintext character, convert it to a key-defined pattern of length five and, using this pattern as a template with one to five holes, select certain ciphertext characters from a 5×5 key-defined grid. (A more complete description can be found in [4].)

Handycipher is based on a core-cipher which operates on plaintext strings over the 31-character alphabet A comprising the 26 uppercase letters {`A-Z`} together with the five symbols {`, . - ? ^`}, and generates ciphertext strings over the 50-character alphabet A' comprising the 50 uppercase and lowercase letters {`A-Y`} and {`a-y`}. Some permutation of these 50 characters plus the space symbol `^` is chosen as a secret shared 51-character key K, as for example:

$$K = \texttt{QjufGCtwbUSNLqHAgVDOoansIhyBKJWFdxvPk\^{}peXMTlirYRmcE}$$

The 50 non-space characters of K (i.e., all but `^`) are displayed as a 5×10 table T_K by writing successive groups of ten characters into the five rows of the table, as, continuing with the example:

$$T_K = \begin{array}{|c|c|c|c|c||c|c|c|c|c|} \hline \texttt{Q} & \texttt{j} & \texttt{u} & \texttt{f} & \texttt{G} & \texttt{C} & \texttt{t} & \texttt{w} & \texttt{b} & \texttt{U} \\ \hline \texttt{S} & \texttt{N} & \texttt{L} & \texttt{q} & \texttt{H} & \texttt{A} & \texttt{g} & \texttt{V} & \texttt{D} & \texttt{O} \\ \hline \texttt{o} & \texttt{a} & \texttt{n} & \texttt{s} & \texttt{I} & \texttt{h} & \texttt{y} & \texttt{B} & \texttt{K} & \texttt{J} \\ \hline \texttt{W} & \texttt{F} & \texttt{d} & \texttt{x} & \texttt{v} & \texttt{P} & \texttt{k} & \texttt{p} & \texttt{e} & \texttt{X} \\ \hline \texttt{M} & \texttt{T} & \texttt{l} & \texttt{i} & \texttt{r} & \texttt{Y} & \texttt{R} & \texttt{m} & \texttt{c} & \texttt{E} \\ \hline \end{array}$$

A 31-plaintext-character sub-key $\bar{K}$ is derived from K by omitting the 20 lowercase letters {`f-y`} and substituting {`Z , . ? -`} for the letters {`a b c d e`}, respectively

$$\bar{K} = \texttt{QGC,USNLHAVDOZIBKJWF?P\^{}-XMTYR.E}$$

and is displayed as a substitution table, $\xi_{\bar{K}}$

$$\xi_{\bar{K}}: \left\{ \begin{array}{ccccccccccccccccccccccccccccccc} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} & \texttt{,} & \texttt{.} & \texttt{-} & \texttt{?} & \texttt{\^{}} \\ 10 & 16 & 3 & 12 & 31 & 20 & 2 & 9 & 15 & 18 & 17 & 8 & 26 & 7 & 13 & 22 & 1 & 29 & 6 & 27 & 5 & 11 & 19 & 25 & 28 & 14 & 4 & 30 & 24 & 21 & 23 \end{array} \right.$$

Then, simply by referring to T_K and $\xi_{\bar{K}}$, plaintext characters are encrypted into k-tuples of ciphertext characters by means of the following scheme:

> Regarding the first five columns of T_K as a 5×5 matrix comprising five rows, five columns, and ten diagonals[1], each plaintext character `m` is encrypted by first expressing $\xi_{\bar{K}}$`(m)` as a five digit binary number $b_1 b_2 b_3 b_4 b_5$ and by using the position of the `1`'s in this number as a pattern, associating the plaintext character `m` with a subset of the ciphertext characters comprising a randomly chosen row, column, or diagonal. Then a randomly chosen permutation of that subset is taken as the corresponding k-tuple of ciphertext characters.
>
> For example, the plaintext character `?` occupying position 21 = 10101 is encrypted into one of the six permutations of one of the twenty 3-tuples
>
> ```
> QuG SLH onI Wdv Mlr QoM jaT unl fsi GIr
> Qnr jsM uIT fol Gai QsT jIl uoi far GnM
> ```
>
> whereas the plaintext character `A` occupying position 10 = 01010 is encrypted into one of the two permutations of one of the twenty 2-tuples
>
> ```
> jf Nq as Fx Ti SW NF Ld qx Hv
> Nx Lv qW HF Sd Hd Sx Nv LW qF
> ```

2 The Core-cipher

This roughly sketched scheme is now defined more precisely as follows. A plaintext message M is encrypted into a ciphertext cryptogram C using a 51-character key K by means of the encryption algorithm E defined as:

Core-cipher Encryption Algorithm: $C \Leftarrow E(K, M)$

> First, omitting ^ the remaining 50 characters of K are displayed as a 5×10 **key-table** T_K by writing successive groups of ten characters into the five rows of the table.
>
> The first five columns of T_K comprise a 5×5 **key-matrix** M_K and the rows, columns, and diagonals of M_K are designated $R_1 - R_5, C_1 - C_5,$ and $D_1 - D_{10}$,

---

[1] The diagonals are extended by "wrapping" around the edges

respectively. We refer to them collectively as **lines**, and call two or more characters **colinear** if they lie in the same line. The 25 characters comprising columns $C_6 - C_{10}$ are said to be **null characters**.

Also, a 31-character plaintext sub-key $\bar{K}$ is derived from K by omitting the 20 lowercase letters {`f-y`} and substituting {`Z , . ? -`} for the letters {`a b c d e`} respectively, and a simple (numerical coding) substitution $\xi_{\bar{K}}$ is applied, transforming each character m of M into the number $\xi_{\bar{K}}(m)$ representing its position in $\bar{K}$ (i.e., if $\bar{K} = p_1 p_2 \ldots p_{31}$ then $\xi_{\bar{K}}(m) = i$ where $m = p_i$).

Then the following four steps are applied in turn to each character m of M.

(1) A random choice is made (with equal probability of each of the 20 possible rows, columns or diagonals) between:
  (1.1) **Column-encryption:** One of the five columns in M_K, say C_j, is randomly chosen (with equal probability),
  or
  (1.2) **Row-encryption:** One of the five rows in M_K, say R_j, is randomly chosen (with equal probability) subject to the following three restrictions, where $\hat{m}$ denotes the character immediately following m in M:
    (1.2.1) $\xi_{\bar{K}}(m) \neq 1, 2, 4, 8, \text{or} 16$, and
    (1.2.2) $\xi_{\bar{K}}(\hat{m}) \neq 2^{5-j}$, if the position of the character $\hat{m}$ in M is an odd number, and
    (1.2.3) $\xi_{\bar{K}}(\hat{m}) \neq 2^{j-1}$, if the position of the character $\hat{m}$ in M is an even number.
  or
  (1.3) **Diagonal-encryption:** One of the ten diagonals in M_K, say D_j, is randomly chosen (with equal probability) subject to the restriction that $\xi_{\bar{K}}(m) \neq 1, 2, 4, 8, \text{or} 16$.

(2) $\xi_{\bar{K}}(m)$ is expressed as a five digit binary number, $b_1 b_2 b_3 b_4 b_5$, and if the position of the character m in M is an odd number, then
  (2.1) If 1.1 was chosen in step 1, then for each i such that $b_i = 1$, the i-th element of C_j is chosen, yielding a subset of the five characters comprising C_j, or
  (2.2) If 1.2 was chosen in step 1, then for each i such that $b_i = 1$, the i-th element of R_j is chosen, yielding a subset of the five characters comprising R_j, or
  (2.3) If 1.3 was chosen in step 1, then for each i such that $b_i = 1$, the i-th element of D_j is chosen, yielding a subset of the five characters comprising D_j.
  But if the position of the character m in M is an even number, then
  (2.4) If 1.1 was chosen in step 1, then for each i such that $b_i = 1$, the $(6 - i)$-th element of C_j is chosen, yielding a subset of the five characters comprising C_j, or
  (2.5) If 1.2 was chosen in step 1, then for each i such that $b_i = 1$, the $(6 - i)$-th element of R_j is chosen, yielding a subset of the five characters comprising R_j, or

(2.6) If 1.3 was chosen in step 1, then for each i such that $b_i = 1$, the $(6 - i)$-th element of D_j is chosen, yielding a subset of the five characters comprising D_j.

(Thus for each successive plaintext character the process alternates between reading rows left-to-right or right-to-left and between reading columns and diagonals top-down or bottom-up.)

(3) The elements of the subset specified in step 2 are concatenated in a randomly chosen order. If this string, composed of 1 to 5 ciphertext characters, satisfies both of the following two restrictions, where $\bar{m}$ denotes the character immediately preceding m in M, then it is taken as $\sigma(m)$. Otherwise, step 1 is restarted.[2]

(3.1) The first character of $\sigma(m)$ must not lie in the line used to encrypt $\bar{m}$, and

(3.2) The first character of $\sigma(m)$ must be colinear with the last character of $\sigma(\bar{m})$, unless $\xi_{\bar{K}}(\bar{m}) = 1, 2, 4, 8, \text{or} 16$ in which case the first character of $\sigma(m)$ must be non-colinear with the single character of $\sigma(\bar{m})$.

(4) So-called **noise characters** are randomly inserted into each $\sigma(m)$ produced in step 3 in the following manner: following each character of $\sigma(m)$ except the first, any one of the eight characters non-colinear with that character[3] may optionally be inserted (i.e., one such character may or may not be inserted).

Finally, the strings produced in step 4 for each character of M are concatenated forming C.

As a result of the restrictions contained in steps 1 and 3, the resulting ciphertext cryptogram C, consisting of the string $\sigma(m_1)\sigma(m_2)\sigma(m_3)\ldots$ can be unambiguously decrypted into the plaintext message $M = m_1m_2m_3\ldots$ by means of the decryption algorithm D defined as follows:

Core-cipher Decryption Algorithm: $M \Leftarrow D(K, C)$

C is divided into contiguous groups of characters, proceeding from left to right, discarding noise characters, at each stage grouping as large an initial segment of the remaining ciphertext as possible composed of mutually colinear characters of

---

[2] It's fairly straightforward to show that some combination of choices made in steps 1 and 3 satisfying all restrictions must exist unless $\xi_{\bar{K}}(m) \cdot \xi_{\bar{K}}(\bar{m}) = 16$ for two consecutive plaintext characters, which would require the two consecutive ciphertext characters to lie in the same row. Accordingly, for each key there will be five bigrams which cannot be encrypted by the algorithm; such bigrams can be handled by hyphenating them.

[3] In order to facilitate visualizing the extended diagonals it's helpful to think of the key-matrix as a 5 × 5 chess board (where the rows, columns, and diagonals wrap around the edges) and recognize that for any given square, 16 of the remaining 24 squares are colinear while just 8 – those that are a knight's move away – are non-colinear.

M_K, then inverting the association between binary numbers and subsets of column, row, or diagonal elements invoked in step 2 of the encryption algorithm, and finally decoding that number by inverting the substitution $\xi_{\bar{K}}$.

Thus each plaintext character m is encrypted by randomly choosing a line of the key-matrix M_K and representing that character's numerical code $\xi_{\bar{K}}(m)$ by an n-tuple $\sigma(m)$ of characters lying in the chosen line, but the randomly chosen line is required to be a column in case $\xi_{\bar{K}}(m) = 1, 2, 4, 8$, or 16. So that in decryption it will be possible to tell where one encrypted character ends and the next begins, $\sigma(m)$ is not allowed to begin with any character lying in the line chosen for $\sigma(\bar{m})$. Noise characters are recognizable because although they lie outside of the line determined by the first two characters of the $\sigma(m)$ being decrypted, being non-colinear with the previous character, they cannot be the beginning of the next encrypted character.

3 Example Encryption With the Core-cipher

Using the key-table T_K from Section 2:

$T_K =$

Q	j	u	f	G	C	t	w	b	U
S	N	L	q	H	A	g	V	D	O
o	a	n	s	I	h	y	B	K	J
W	F	d	x	v	P	k	p	e	X
M	T	l	i	r	Y	R	m	c	E

and its associated substitution table, ξ_K:

$\xi_{\bar{K}}$:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	,	.	-	?	^
10	16	3	12	31	20	2	9	15	18	17	8	26	7	13	22	1	29	6	27	5	11	19	25	28	14	4	30	24	21	23

the plaintext CATS AND DOGS could be encrypted as follows:[4]

---

[4] In the third column $\xi_{\bar{K}}(m)$ is expressed in binary; in the fourth column the row, column, or diagonal chosen in Step 1 is indicated.

m	$\xi_{\bar{K}}(m)$		`C/R/D`	$\sigma(m)$	$\sigma(m)$+`noise`
`C`	3	00011	`R5`	`ri`	`rin`
`A`	10	01010	`R2`	`qN`	`qN`
`T`	27	11011	`R4`	`xFvW`	`xFvqW`
`S`	6	00110	`C3`	`Ln`	`Ln`
`ˆ`	23	10111	`D10`	`GnMF`	`GnMF`
`A`	10	01010	`D1`	`Nx`	`Nx`
`N`	7	00111	`D6`	`sdT`	`sdT`
`D`	12	01100	`C2`	`Fa`	`Fa`
`ˆ`	23	10111	`D4`	`oFfl`	`oFLfNl`
`D`	12	01100	`C4`	`xs`	`xs`
`O`	13	01101	`D1`	`nNr`	`nNr`
`G`	2	00010	`C3`	`L`	`L`
`S`	6	00110	`C2`	`Fa`	`Fa`

yielding the ciphertext:

`rinqNxFvaWLnGnMFNxsdTFaoFLfNlxsnNrLFa`

This ciphertext would be decrypted by scanning it from left to right, deleting noise characters, and dividing it, according to the table T_K, into its constituent k-tuples and then finding each group's associated binary number, converting to decimal, and decoding by inverting the substitution ξ_K

$$\xi_{\bar{K}}^{-1}: \left\{ \begin{array}{ccccccccccccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ \texttt{Q} & \texttt{G} & \texttt{C} & \texttt{,} & \texttt{U} & \texttt{S} & \texttt{N} & \texttt{L} & \texttt{H} & \texttt{A} & \texttt{V} & \texttt{D} & \texttt{O} & \texttt{Z} & \texttt{I} & \texttt{B} & \texttt{K} & \texttt{J} & \texttt{W} & \texttt{F} & \texttt{?} & \texttt{P} & \texttt{\^{}} & \texttt{-} & \texttt{X} & \texttt{M} & \texttt{T} & \texttt{Y} & \texttt{R} & \texttt{.} & \texttt{E} \end{array} \right.$$

Thus, `rin` is recognized as the bigram `ri` because `n` is non-colinear with `i` (and so is a noise character) while the following `q` is colinear with `i` but lies outside the line determined by `ri`, and `ri` is associated with 00011 = 3 which is decoded as `C`. Then `qN` is recognized as the bigram `qN` because the following `x` is colinear with `N` but lies outside the line determined by `qN`, and `qN` is associated with 01010 = 10 which is decoded as `A`. Then `xFvqW` is recognized as the 4-tuple `xFvW` because `q` is non-colinear with `v` (and so is a noise character) while the following `L` is colinear with `W` but lies outside the line determined by `xFvW`, and `xFvW` is associated with 11011 = 27 which is decoded as `T`, and so forth.

For a slightly larger example consider the 229-character plaintext:

```
IT HAUNTS ME, THE PASSAGE OF TIME. I THINK TIME IS A MERCILESS THING. I THINK LIFE
IS A PROCESS OF BURNING ONESELF OUT AND TIME IS THE FIRE THAT BURNS YOU. BUT I
THINK THE SPIRIT OF MAN IS A GOOD ADVERSARY. -- TENNESSEE WILLIAMS[5]
```

which can be encrypted by the Core-cipher as, for example, this 873-character ciphertext:

---

[5] where ˆ characters have been replaced by spaces for legibility.

```
isWxq LfdWr lMriH udfTi aGnld HSNqx soHlf LWflr MiTSs FdIvW MSWQf xWoTQ SnMTs rQLHL
STqFN nFNLf jSriT MlouM iFNnN rTaGQ fuFiq xGfrN fxnsa HIFSW lMoGQ LGjqu fiGad ILjvs
ndflQ uWusT nqMLu dMLnx sNiMQ LGuSQ fLMuv jqlFo HqxiM FGnfq lNjaT MLvsI qFGLM nvasi
adGrT invNu MoTiT jWaFv iLGqd FnLQu Gsfja FnaHi sfxGW fLIrl SdHjQ saFTq dunLH folLl
Qdnan oIxrN QjxuF nMqSI jflqr fLaSs aoGnQ ondLG ulIxj lauIL Wqsdu Lolnj qNiqs fliMo
uxivL lnMTH TdfsQ NLvsT qIuiS anMos FnIlv dGniM Fdxvo WQISa WrNsv jMnGF iWoMN lusLQ
ndFnf NvnoG iLuLT IoWMu QjsWv HLdHQ aTxuf SGsQH Fxsio WFofl GxrNQ TdHas iSdru laLGn
dMaiQ SnGnl dLaSQ Gurfl LuHdG jMLsv joxdW svITN xnrGu fQWSM ojQnG TqMSM WjQjx vNTir
oMnMF GinrT aFsjM vfsro Hvdfx WfGuM NLFHo uFQWS oIjlG SQafl xHSja fruLH qLNif qsQNx
rGdFN alTHo fulHT QsMvL jFfaG SiFdu jQiGa dsiqS xGIlQ iaGdL WlraI soaid IaqGn dufoQ
ujiGd QFHvN xiIsT sSQHo WSMdq ISqHn LvNQr nxqaS ifoqF inaGj oQiuN qxdLd lTiQr lMLNS
HsqNv xsfiQ lTvML svMdr xQInj oSQMq fdrWu GfQLT riQMj aNfFT uNons Iqisx fWaTl xvdfF
WnflH oFMru ilSGi vsMHL WNsdQ HqNFM qWI
```

parsed as:

```
I     T     ^     H   A  U  N   T    S  ^    M   E      , ^     T    H
isWxq LfdWr lMriH udf Ti aG nld HSNq xs oHlf LWf lrMiTS s FdIvW MSWQ fx

E       ^      P    A  S  S  A  G E     ^      O   F  ^     T     I
WoTQSnM TsrQLH LSTq FN nF NL fj S riTMl ouMiFN nNr Ta GQfuF iqxGf rNfxn

M     E        .     ^     I    ^      T      H  I     N    K   ^
saHIF SWlMoGQL Gjquf iGadI Ljvs ndflQu WusTnq ML udMLn xsNi MQL GuSQf

T      I    M   E       ^    I    S  ^      A  ^    M    E       R
LMuvjq lFoH qxi MFGnfql NjaT MLvs Iq FGLMnv as iadG rTin vNuMoTi TjWaF

C   I     L E      S  S   ^     T     H  I     N   G .    ^    I
viL GqdFn L QuGsfj aF naH isfxG WfLIr lS dHjQs aFT q dunL Hfol LlQdn

^    T    H   I    N   K   ^    L I     F  E      ^     I     S  ^
anoI xrNQ jxu FnMq SIj flq rfLa S saoGn Qo ndLGul Ixjla uILWq sd uLolnj

A  ^    P   R     O   C  E      S  S  ^    O   F  ^     B U  R     N
qN iqsf liM ouxiv Lln MT HTdfsQ NL vs TqIu iSa nM osFnI l vd GniMF dxv

I     N   G ^    O    N    E       S   E        L F  ^     O    U
oWQIS aWr N svjM nGFi WoMN lusLQnd Fnf NvnoGiLu L TI oWMuQ jsWv HL

T      ^      A  N   D  ^     T    I     M    E        ^      I    S
dHQaTx ufSGsQ HF xsi oW FoflG xrNQ TdHas iSdr ulaLGndM aiQSnG nldL aS

^     T     H  E      ^     F  I    R    E      ^     T     H  A  T
QGurf lLuHd Gj MLsvjo xdWsv IT Nxnr GufQ WSMojQ nGTqM SMWjQ jx vN TiroM

^    B U  R     N     S  ^     Y    O    U   .    ^     B U  T
nMFG i nr TaFsj Mvfsr oH vdfxW fGuM NLFH ouF QWSo IjlGS Q af lxHSj

^     I    ^    T    H  I     N   K  ^    T    H  E      ^     S  P
afruL HqLN ifqs QNxr Gd FNalT Hof ul HTQs MvLj Ff aGSiFd ujQiG ad siq
```

```
I      R    I     T    ^      O   F  ^     M    A  N    ^     I     S
SxGIlQ iaGd LWlra Isoa idIaqG ndu fo QujiG dQFH vN xiIs TsSQH oWSMd qI

^     A  ^    G O   O     D  ^      A  D  V   E      R    S  A  R
SqHnL vN Qrnx q aSi foqFi na GjoQiu Nq xd Ldl TiQrlM LNSH sq Nv xsfiQ

Y    .     ^      ^    -   -  ^     T     E      N   N   E     S  S
lTvM LsvMd rxQInj oSQM qfd rW uGfQL TriQM jaNfFT uNo nsI qisxf Wa Tl

E       E     ^     W   I     L L I    A  M   S
xvdfFWn flHoF Mruil SGi vsMHL W N sdQH qN FMq WI
```

4 Handycipher

While the Core-cipher itself has proven to be remarkably robust when encrypting relatively short plaintexts (less than a few hundred characters), with increasing message length it becomes more vulnerable to statistically based hill-climbing attacks along the lines described by Dhavare, et al.[3] even though the random insertion of noise characters is designed to interfere with such attacks. Therefore Handycipher is further strengthened by randomly salting the ciphertext with additional decoys chosen from the 25 null characters of columns C_6–C_{10} of the key-matrix.

After a message has been encrypted by the Core-cipher using the key K into a ciphertext C' (composed of non-nulls), a slightly longer string N of nulls is generated by randomly choosing from K's 25 nulls, and then N is randomly interleaved with C' by means of the following procedure.

Salting Algorithm: $R \Leftarrow$ salt(S,T)

Given two strings $S = s_1 s_2 \ldots s_{|S|}$ and $T = t_1 t_2 \ldots t_{|T|}$, S is said to be **salted** with T by randomly interleaving the two strings, resulting in a string $R = r_1 r_2 \ldots r_{|R|}$ constructed by the following process, which is designed to incorporate all of S while randomly interleaving successive characters of T (repeating from the beginning of T in case the end of T is reached before S is exhausted).

```
$i \leftarrow 0$      $j \leftarrow 0$      $k \leftarrow 0$
WHILE $j \le |S|$
    flip a fair coin
    IF heads
        increment $i$
        increment $j$
        IF $j \le |S|$
            $r_i \leftarrow s_j$
        ENDIF
    ELSE
        increment $i$
        increment $k$
        $r_i \leftarrow t_k$
        IF $k = |T|$
            $k \leftarrow 0$      (Also issue a warning that $T$ should be made longer.)
        ENDIF
    ENDIF
    IF $i > 1$ and $r_i = r_{i-1}$
        $i \leftarrow i - 1$      (This deletes any duplicates from R.)
    ENDIF
ENDWHILE
```

So that the interleaved null characters will statistically resemble the non-nulls, they are chosen in a manner designed to make close repetition of identical nulls unlikely. Also, in order to thwart a known attack based on exploiting the absence of duplicate characters in the ciphertext, Handycipher uses a variant form $\tilde{E}$ of the Core-cipher encryption algorithm $C \Leftarrow E(K,M)$ in which the definition of *noise character* is broadened to include duplicate characters by amending Step (4) of the Core-cipher Encryption Algorithm as follows:

> (4) So-called **noise characters** are randomly inserted into each $\sigma(m)$ produced in step 3 in the following manner: following each character of $\sigma(m)$ except the first, ***either a duplicate of that character, or*** any one of the eight characters non-colinear with that character may optionally be inserted (i.e., one such character may or may not be inserted).

To accomplish all of the above, Handycipher is defined as the following extension of the Core-cipher:

Handycipher Encryption Algorithm: $\boldsymbol{C \Leftarrow E^{\dagger}(K, M)}$

1. $C' \Leftarrow \tilde{E}(K, M)$
2. Generate a somewhat longer string N of nulls[6] by sequentially choosing nulls at random from the set of 25, but potentially rejecting each choice with probability $(6-k)/6$ if that choice would duplicate the kth previous choice, for $1 \leq k \leq 5$.[7]
3. $C \Leftarrow \mathrm{salt}(C', N)$[8]

and the corresponding decryption is simply accomplished as:

Handycipher Decryption Algorithm: $\boldsymbol{M \Leftarrow D^{\dagger}(K, C)}$

> This algorithm is identical to the Core-cipher decryption algorithm except that the phrase *discarding noise characters* is amended to read *discarding noise and null characters.*

5 Extended Handycipher

Extended Handycipher operates with the same plaintext and ciphertext alphabets, and encrypts a message M using a key K by first generating a random session key K', and encrypting M with Handycipher using K' to produce an intermediate ciphertext C'. K' is then encrypted with Handycipher using K, and then embedded in C' at a location based on K and the length of M, producing the final ciphertext C.

Extending Handycipher in this way confers several advantages in security at little computational cost. Because each plaintext message is encrypted with a different randomly generated session key, the primary secret key is less exposed to any attack that depends on having a lot of ciphertext to work with, and the security of the cipher is less compromised by encrypting multiple messages with the same key.

Extended Handycipher Encryption Algorithm: $\boldsymbol{C \Leftarrow E^{*}(K, M)}$

1. Generate a random 51-character **session key** K' with associated table $T_{K'}$ and coding substitution $\xi_{\bar{K}'}$.
2. Transcribe K' into plaintext characters by inserting a "`-`" before each run of uppercase letters and a "`.`" before each run of lowercase letters, and then changing all lowercase letters to uppercase.[9]

---

[6] The exact number of nulls called for by the salting algorithm will depend on K, M, and C' but typically will not exceed the length of C' by more than 12.5%. In case N is too short, the algorithm will recycle it from the beginning (which introduces a vulnerability) and issue a warning but, of course, when encrypting by hand the required nulls can be selected as needed rather than in advance as the null-string N.

[7] This lessens the probability of identical nulls occurring close to each other in N.

[8] Note that all the duplicates introduced in Steps 1 and 2 are removed from C by the Salting Algorithm.

[9] E.g., the Section 2 key `QjufGCtwbUSNLqHAgVDOoansIhyBKJWFdxvPk^peXMTlirYRmcE` is transcribed as `-Q.JUF-GC.TWB-USNL.Q-HA.G-VDO.OANS-I.HY-BKJWF.DXV-P.K^PE-XMT.LIR-YR.MC-E`

3. Encrypt the transcribed K' with Handycipher and K, yielding K''. Adjust K'' if necessary ensuring that for the last character m of the transcribed K' to be encrypted, no null characters are interspersed with $\sigma(m)$ and that K'' terminate with exactly one null character.[10]
4. Encrypt M with Handycipher and K', yielding C'.
5. Adjust C' if necessary, by inserting more nulls, ensuring that $|C'| + |K''| \geq 700$ and also that $N \geq 30 - R$ where $|C'| = 31 \cdot N + R$, $0 \leq R < 31$.
6. Calculate $j = \lfloor (|C'| + |K''| - 700)/31 \rfloor \cdot \{[\xi_{\bar{K}}(\mathtt{A}) + \xi_{\bar{K}}(\mathtt{B}) + \xi_{\bar{K}}(\mathtt{C})] \pmod{31}\} + [\xi_{\bar{K}}(\mathtt{D}) + \xi_{\bar{K}}(\mathtt{E}) + \xi_{\bar{K}}(\mathtt{F})] \pmod{31}$.[11]
7. Insert K'' into C' immediately following position j as calculated in step 6, yielding C.

Extended Handycipher Decryption Algorithm: $M \Leftarrow D^*(K, C)$

1. Calculate $j = \lfloor (|C| - 700)/31 \rfloor \cdot \{[\xi_{\bar{K}}(\mathtt{A}) + \xi_{\bar{K}}(\mathtt{B}) + \xi_{\bar{K}}(\mathtt{C})] \pmod{31}\} + [\xi_{\bar{K}}(\mathtt{D}) + \xi_{\bar{K}}(\mathtt{E}) + \xi_{\bar{K}}(\mathtt{F})] \pmod{31}$ and begin decrypting the substring of C immediately following position j with Handycipher and K.
2. Transcribe all letters between a "`.`" and a following "`-`" into lowercase and then delete all occurrences of those two symbols.
3. Continue until 51 such characters have been decrypted, yielding the session key, K'.
4. Remove the decrypted substring from C, leaving C'.
5. Decrypt C' with Handycipher and K', yielding M.

6 Version History

The earliest versions posted in the *IACR Cryptology ePrint Archive* (April, 2014 v1.3 and July, 2014 v2.1) quickly succumbed to hill-climbing attacks predicated on the ability to discover and remove the very small set of only five nulls.[1][2]

A subsequent version (December, 2014 v4.4) addressed this weakness by adding ten characters to the ciphertext alphabet, using a 41-character key instead of 31, increasing the number of null characters from five to fifteen, increasing the number of diagonals used from two to ten, and alternating the direction of encoding plaintext characters between top-down/left-right and bottom-up/right-left. That version was the basis of the previous MTC3 challenges Handycipher Parts 1–3, Extended Handycipher Parts 1–3, and Weakened Handycipher Parts 1–3.

---

[10] This is necessary so that in Step 3 of the decryption algorithm the end of K'' can be recognized.

[11] Here $\lfloor x \rfloor$ denotes the integer part of x and $|C|$ denotes the length of C. The formula is designed merely to make the value of j depend on K (and it's sub-key $\bar{K}$) and $|C|$. The adjustments in Step 5 ensure that $j \leq |C|$.

The cipher was further strengthened (April, 2016 v6.8) by adding another ten characters to the ciphertext alphabet, using a 51-character key instead of 41, increasing the number of null characters from 15 to 25, and by interweaving random non-null "noise" characters in the Core-cipher encryption before the null characters are added. That version was the basis of the previous MTC3 challenges Handycipher Parts 4–6, Extended Handycipher Parts 4–6, and Weakened Handycipher Parts 4–6.

The current version (November, 2016 v6.10) represents a slight change removing a subtle vulnerability which could be exploited to facilitate identification of the nulls. MTC3 challenges Handycipher Parts 7–9 are based on this version.

References

1. S. Combes, Handycipher Decrypt (2014), available at http://oilulio.wordpress.com/2014/06/19/handycipher-decrypt/
2. S. Combes, Breaking Handycipher 2 (2014), available at http://oilulio.wordpress.com/2014/07/28/breaking-handycipher-2/
3. A. Dhavare, R. M. Low, M. Stamp, Efficient cryptanalysis of homophonic substitution ciphers, Cryptologia 37 (2013) 250-281
4. B. Kallick, Handycipher: a Low-tech, randomized, symmetric-key cryptosystem (2014), available at http://eprint.iacr.org/2014/257.pdf