Security of PassRules™

James D. Currie
University of Winnipeg

The PassRules™ protocol for user identification gives a level of security much higher than that of PIN-based systems in that the user enters a different PIN on the keypad every time she makes a transaction. Several variations of the protocol offer very high entropies. This report focuses on evaluating the statistical and information theoretical aspects of the protocol.

Executive Summary

Overview

PassRules™ is a patented protocol for user identification. Instead of remembering a fixed PIN, a user remembers a sequence of cells from a 6×6 grid of digits. In response to a challenge grid, presented by a terminal, digits appearing in the remembered cells seed the formation of a passcode of 4 or more digits.

Security analysis

1. Section 2 of this report gives the entropies of various protocols. We consider several versions of PassRules™ grouped according to ease of memorization and the lengths of the resulting passcodes (PINs). In particular, all versions of PassRules™ are seen to be far superior in this regard to standard 4 or 5-digit PINs.

   In the case of the wizard-style PassRules, the greatest contribution to entropy is shown to come from **pair style** PassRules, so the properties of these rules are particularly important.

Protocol	Number of sub-rules	Length of passcode	Entropy
PIN	n/a	4	13.29
extended PIN (92 char keyboard)	n/a	4	26.09
password on $\{A–Z, 0–9\}$	n/a n/a	6	31.02
password on $\{A–Z, 0–9\}$	n/a n/a	8	41.36
PassRules™ (unrestricted)	4	4–8	47.23
PassRules™ (unrestricted, chopping)	4	4	47.23
PassRules™ (restricted)	4	4–8	43.12
PassRules™ (restricted, chopping)	4	4	43.12
PassRules™ (unrestricted)	4	4	43.96
PassRules™ (restricted)	4	4	41.42
PassRules™ (restricted)	4	4–5	42.13
PassRules™ (unrestricted)	5	5–10	59.03
PassRules™ (unrestricted, chopping)	5	5	59.03
PassRules™ (restricted)	5	5-10	52.39
PassRules™ (restricted, chopping)	5	5	52.39
PassRules™ (restricted)	5	5	50.69
PassRules™ (restricted)	5	5–6	51.25

Figure 1: Entropies for various PassRules™ protocols. Here a 'restricted' PassRule follows a format designed to allow its easy memorization.

dimensions	operators	entropy
2×2	unary	17.04
2×3	unary	18.36
2×4	unary	19.04
2×4	binary	15.97
2×5	binary	16.07
2×6	binary	14.54
total		21.15

Figure 2: Entropies for PassRules in array style

length n	entropy
4	17.26
5	18.40
6	19.20
7	19.82
8	20.29
9	20.65
10	20.95
11	21.22
12	21.50
13	21.73
14	21.93
15	22.12
16	22.29
all lengths	24.81

Figure 3: Entropies for PassRules in line style

quadrant cells	operators	entropy
1	unary	17.19
2	unary	22.77
2	binary	18.48
4	binary	25.26
all types		25.51

Figure 4: Entropies for PassRules in quadrant style

Number of Pairs	Entropy
4	26.78
5	31.81
6	36.52
7	40.95
8	45.10
total	45.18

Figure 5: Entropies for PassRules in pair style

2. Section 3 considers the ease with which the correct passcode response to a given grid may be guessed. It is shown that when challenge grids are generated correctly, the probability of guessing a passcode is essentially no greater than that of guessing a random PIN. When we attempt to lower the chance of guessing a PIN there is a cost in increased 'entropy leakage'.

3. Section 4 examines security of PassRules™ in those cases where a grid/response pair is observed. We first (Section 4.1) consider phishing schemes, where a user is duped into responding to tailored grids chosen by an adversary. PassRules™ is more robust in response to such attacks than PINs; however, as for any password style protocol, phishing remains a dangerous attack. The trivial, non-altering versions of PassRules™ would be vulnerable to phishing. This vulnerability also exists (to a lesser extent) for unary PassRules, such as those of line style. We also sketch ideas for a phishing scheme targeting pair style PassRules. The existence of phishing schemes means that the eventual implementation of PassRules should consider carefully how many times users will be prompted for a PIN after an incorrect code entry.

4. Section 4.2 examines the security of PassRules™ in those cases where random grid/response pairs are observed. Because of its high entropies, PassRules™ does considerably better here than PINs. However, in the case of trivial PassRules (but also in the case of pair style PassRules!) most of the entropy disappears after 2 grid/response pairs are observed. This may indicate that even more entropy should be added to the way that rules can be generated.

Conclusion

We have given an analysis of several aspects of PassRules™. This has involved the calculation of many entropies. We have analyzed multiple variations of PassRules™ and have identified an important issue, namely the tradeoff between passcode guessing and entropy leakage. The security that PassRules™ offers against unauthorized logins or card fraud will be better than that of ordinary chip and pin. PassRules™ will be stronger than PINs with respect to shoulder surfing and entropy leakage. However, no security system gives absolute safety, and PassRules is not invulnerable to phishing and entropy leakage. It seems that these problems are worse when PassRules are highly structured. More flexibility in the way 'memorable" PassRules that are assigned to users are generated may be necessary.

Contents

1 Introduction

PassRules™ is a patented protocol for user identification. Instead of remembering a fixed PIN, a user remembers a sequence of cells from a 6×6 grid of digits. The digits in these cells are used as input for operators, which are applied to give a passcode.

1.1 Description of PassRules™

In a 6×6 grid, the cells may be labelled as $(1), (2), \ldots (36)$. Operators which are applied to the cell contents are of two types: binary $(+, \times, -, b, s)$ and unary $(+0, +1, +2, +3, +4, +5, +6)$. A **PassRule** consists of a sequence of these operators, together with a specification of the cells to which operators are applied. Each subsequence consisting of an operator and its cells is referred to as a **sub-rule**.

The operators of PassRules™ are chosen to be simple. A sub-rule of the form $(i)+j$ returns the result of adding j to the contents of cell i. A sub-rule of the form $(i)*(j)$ where b is a binary operator returns the result of applying operator $*$ to the contents of cells i and j. Operators $+$ and $\times$ are the usual addition and multiplication, while if the contents of cells i and j are x and y respectively, then the results are:

$$\begin{aligned} (i) - (j) &\rightarrow |x - y| \\ (i) b (j) &\rightarrow \max(x, y) \end{aligned}$$

$$(i)s(j) \quad \rightarrow \quad \min(x, y).$$

The sequence of sub-rule results, taken together, form the **passcode** or PIN corresponding to a PassRule and grid.

A weaker version of PassRules™ has been studied The grid for this protocol is 5×5, and no operators are used. Alternatively, one could say that the only operator is $+0$.

1.2 Previous work

Various internal technical reports on PassRules™ [5, 3, 4] exist, exploring syntax, entropies and specifications for a PassRule-generating 'wizard'. A trivial version of the protocol has already received some analysis [9, 2].

1.3 Double digit sub-rule results

One wrinkle of the PassRules™ protocol is that a given rule may produce passcodes of differing lengths. To give the simplest example, the rule $(1)+1$ gives a one-digit passcode when (1) is 8 or less, but the two-digit code **10** when (1) is 9. This means that from the observation of a single passcode, an adversary will be unable to determine the number of sub-rules in a user's PassRule. However, this property of PassRules™ can also give away information. If a user's passcodes are observed or captured twice and seen to have different lengths, then his PassRule must use one of the operations $+1, +2, +3, +4, +5, +6, +$ or $\times$. It seems to us that avoiding giving away information about operators is more compromising than giving away information about length; in a standardized setting, the length of users' PassRules will likely already have some default value. Guessing the number of sub-rules would then be a non-issue.

How then should one face this property of PassRules™? Several possibilities have been discussed:

- One could simply return 1-digit or 2-digit results as they arise. We call this option **natural**.
- Since a challenge grid is only ever sent to a user whose PassRule is known, the challenge grid itself could be tailored so that its entries, in

combination with the known PassRule, never give a 2-digit result. Call this option **grid tailoring**.

- One could equally guarantee that every operator returns two digits by using leading 0's:

$$8s6 = 6 \to 06, \quad 5 + 2 = 7 \to 07.$$

  Call this option **padding**

- One could guarantee that every operator returns a single digit by keeping only the last digit of a 2-digit code. This is equivalent to keeping the remainder after dividing by 10. For example

$$5 + 8 = 13 \to 3, \quad 9 \times 5 = 45 \to 5.$$

  Call this option **chopping**.

As we have mentioned, the natural option definitely gives away information concerning operations in sub-rules. There will also be entropy leakage associated with the grid tailoring option. For example, suppose a sub-rule uses the operator +6. If we avoid the 'double digit problem' by ensuring that the cell to which +6 is applied never causes a 2-digit sum, then the cell's contents are always 0, 1, 2 or 3. An observer seeing several challenge grids corresponding to the given PassRule can detect this regularity – without even needing to capture the passcodes entered in response to the grids. There is a definite loss of entropy associated with this solution to the double digit property.

The third suggested solution to the double digit problem is to pad all single digit results from a sub-rule. Thus 6 becomes 06, 2 becomes 02, etc. Again, in this solution we give away information about sub-rules and cells. A PassRule which gives a passcode ending in, for example 56, must have computed 7×8. A PassRule ending in 24 comes from either 3×8 or 4×6. By way of comparison, if we use the 'chopping' protocol, a sub-rule generating 56 will have its result truncated to 6, which can arise in many, many ways ($1 + 5$, $9 - 3$, 1×6, 2×3, 2×8, $2b6$, etc.)

We see that the least information about sub-rules is given away by the chopping option. The length of a PassRule is not hidden in this option, but we regard transparency in this aspect as relatively unimportant.

2 Entropies

An important property of a protocol for user identification is the total number of valid keys or passwords available. The **entropy** of a key is the base-2 logarithm of the number of guesses necessary to guarantee finding it. Thus, entropy is high exactly when there are a large number of possible keys, and when entropy increases by 1, a key becomes twice as hard to guess. To analyze PassRules™ from this point of view, we consider variations of it which are close to existing PIN schemes; in particular we will count some possible PassRules giving passcodes of 4, 5 or 6 digits. We also count entropies associated with various 'wizard-generated' PassRules.

2.1 4 sub-rules

A 4-digit PIN will be one of the 10^4 numbers from 0000 to 9999. Thus 4-digit PINs have an entropy of

$$\log_2(10000) \approx 13.29.$$

A 4-digit PIN (arguably) corresponds to a PassRule with 4 sub-rules. To specify an arbitrary PassRule with 4 sub-rules:

1. We choose i sub-rules to be unary and $4 - i$ sub-rules to be binary, where $0 \leq i \leq 4$.

2. We choose the unary sub-rules from the 7 possibilities $(+0, +1, +2, +3, +4, +5, +6)$ and the binary sub-rules from the 5 possibilities $(+, -, \times, b, s)$.

3. For each unary rule we choose one cell, while for each binary rule we choose two cells.

The total number of possible PassRules with 4 sub-rules is thus

$$\sum_{i=0}^{4} \binom{4}{i} 7^i \ 5^{4-i} \ 36^i \ 36^{2(4-i)} = 2,053,886,447,390,976.$$

The entropy of PassRules with 4 sub-rules is thus

$$\log_2 2,053,886,447,390,976 \approx 50.87.$$

Guessing a PassRule thus is about 2^{38} times harder than guessing a 4-digit PIN.

By way of comparison, the entropy of 4-cell a sequence of 4 cells from a 5×5 grid is

$$\log_2 25^4 \approx 18.58.$$

The entropy of PassRules with 4 sub-rules is thus seen to be extremely large. Various restrictions to make PassRule memorization easier have been proposed. In fact, which PassRules are easier to remember is an empirical question, perhaps best determined by psychological studies. However, the following restrictions are mentioned in [3]:

1. PassRules should use exclusively unary operators or exclusively binary operators.
2. PassRules should use at most two distinct operators.
3. PassRules should have at most two operator transitions.
4. PassRules should not reuse cells.

Under these **memorization restrictions**, one finds that the number of possible PassRules with 4 sub-rules is

$$152, 512, 479, 753, 480$$

giving an entropy of

$$\log_2 152, 512, 479, 753, 480 \approx 47.12.$$

This is a drop in the entropy. Nevertheless, entropy remains huge relative to 4-digit PINs. Note that some of these PassRules with 4 sub-rules may sometimes give passcodes of lengths 4, 5, 6, 7 or 8 under the natural option, depending on the grid. If, however, chopping is adopted, as per our recommendation in the last section, all these PassRules give 4-digit PINs.

2.1.1 4 sub-rules giving passcodes with exactly 4 digits

If we insist on the 'natural' approach to sub-rule results, then a PassRule with 4 sub-rules can only be guaranteed to give a passcode of length exactly 4 (regardless of the choice of grid) if the sub-rules are chosen from among $\{+0, b, s, -\}$. If we choose PassRules without regard to the memorization

restrictions mentioned in the last section, the number of PassRules with 4 sub-rules guaranteeing a passcode of length exactly 4 is

$$\sum_{i=0}^{4} \binom{4}{i} 3^{4-i}\ 36^{i}\ 36^{2(4-i)} = 237,091,505,746,176.$$

giving an entropy of

$$\log_2 237,091,505,746,176 \approx 47.75$$

On the other hand, if we insist on all 4 memorization restrictions, the number of possible PassRules is reduced to

$$47,583,780,856,920$$

giving an entropy of

$$\log_2 47,583,780,856,920 \approx 45.44.$$

We see that insisting on the memorization constraints does not unduly impact the entropy, so from now on we will mostly consider 'memorable' PassRules obeying these constraints.

2.1.2 4 sub-rules giving passcodes with 4 or 5 digits

As mentioned earlier, by chopping we can force all sub-rules to return a single digit, or by padding we can force all sub-rules to return two digits. These options then cause a PassRule with 4 sub-rules to give either 4 digits (always) or 8 digits (always). Neither these two approaches nor the basic PassRules protocol will force a 4 sub-rule PassRule to always yield a 5-digit passcode. This could, however, be ensured by grid tailoring (which, however, we do not recommend.) In this section we will count those PassRules with 4 sub-rules with results interpreted naturally yield either a 4-digit or 5-digit passcode. This means that at most one 'special' sub-rule is allowed to potentially return a 2-digit number.

We have already counted PassRules with 4 sub-rules interpreted naturally which are guaranteed to give passcodes of length 4. Let us now see how many are added if we allow passcodes of length 5. We choose at most one 'special' sub-rule from among $\{+1, +2, +3, +4, +5, +6, +, \times\}$. Because of the memorization restrictions, we either choose exactly one sub-rule from

$\{+1, +2, +3, +4, +5, +6\}$ and the other sub-rules are all $+0$, or we choose exactly one sub-rule from among $\{+, \times\}$, while the 3 others are some one operator chosen from $\{b, s, -\}$. In either case there are 4 places in the Pass-Rule operator sequence where the single 'special' operator can be placed. With three operators the same, we automatically will have 1 or 2 transitions in the operator sequence. The number of PassRules is thus

$$4\left(6\binom{36}{4}4! + 2 \times 3\binom{36}{8}8!\right) = 29,282,359,740,480.$$

The number of 'memorable' PassRules with 4 sub-rules returning passcodes of length either 4 or 5 is

$$47,583,780,856,920 + 29,282,359,740,480 = 76,866,140,597,400$$

giving an entropy of 46.13.

2.2 5 sub-rules

A 5-digit PIN will be one of the 10^5 numbers from 00000 to 99999 so that 5-digit PINs have an entropy of

$$\log_2(100000) \approx 16.61.$$

By a similar calculation to that in the previous section, the total number of (not necessarily memorable) PassRules with 5 sub-rules is

$$13,826,763,563,836,050,432.$$

with an entropy of

$$\log_2 13,826,763,563,836,050,432 \approx 63.58.$$

Under the memorization restrictions, one finds that the number of possible PassRules is

$$189,090,638,242,894,080$$

giving an entropy of

$$57.39.$$

2.2.1 5 sub-rules giving passcodes with exactly 5 digits

As per our earlier discussion, taking into account the memorization restrictions, possible PassRules with 5 sub-rules giving passcodes with exactly 5 digits is

$$58, 110, 775, 617, 565, 440$$

with corresponding entropy

$$55.69.$$

2.2.2 5 sub-rules giving passcodes with 5 or 6 digits

In this section we count all PassRules yielding either a 5-digit or 6-digit passcode. This means that at most one sub-rule is allowed to potentially return a 2-digit number. If we insist on the memorization restrictions, the number of PassRules is

$$58, 110, 775, 617, 565, 440 + 27, 671, 799, 248, 755, 200 = 85, 782, 574, 866, 320, 640$$

with an entropy of about

$$56.25$$

2.3 Wizard generated sub-rules

Usually in PIN or password schemes, security is heightened when users must use assigned passwords. This can create a problem with users struggling to remember their random passwords. In [4], various specifications are given for wizard-generated PassRules structured to facilitate user memorization. Several PassRule styles are discussed in [4]:

- array style
- line style
- quadrant style
- pair style

The wizard-generated PassRules specify the cells to which operators are applied. In some wizard-generated styles, the types of operators to be used are specified (for example, only binary operators are used in pair style.) The

styles are not disjoint; for example, those all array style PassRules where binary operators are allowed are also pair style PassRules. Both the line and pair styles also overlap with quadrant style.

2.3.1 Operator sequences

In the case of wizard-generated rules we can separate the choice of operator sequence from the choice of cell sequence. Once we are given n cells in sequence, we may count in how many ways n unary operators may be applied to the cells. In the case where binary operators are used, we apply n binary operators to a sequence of $2n$ cells.

We deal first with the unary case: Suppose that we are choosing unary operators to apply to a sequence of n cells. If we use a single unary operator (7 choices), then the sequence of operators is fixed; we apply that single operator to all n cells. If we use two unary operators, we choose a first operator (7 choices) and a second (6 choices) and the intercell boundaries at which the transitions occur (and there are $\binom{n}{2}$ choices.) The total number of ways of applying unary operators to a sequence of n cells is thus

$$7 + 42\binom{n}{2}. \tag{1}$$

In the binary case we are choosing n operators to apply to a sequence of $2n$ cells. If we use a single binary operator, then the sequence of operators is fixed; we apply that single operator to all n pairs. If we use two binary operators, we choose a first operator (5 choices) and a second (4 choices) and the interpair boundaries at which the transitions occur (again there are $\binom{n}{2}$ choices.) The total number of ways of applying binary operators to a sequence of n pairs is thus

$$5 + 20\binom{n}{2}. \tag{2}$$

2.3.2 Array Style

The **array style** wizard-generated PassRules take cells from a rectangular sub-block or **array** of the grid. The cells of the array are sequenced in one of eight ways, specifying left-to-right or right-to-left and top-to-bottom or bottom-to-top, and an order, for example:

- left-to-right, then top-to-bottom (the normal reading sequence for English)
- right-to-left, then top-to-bottom (the normal reading sequence for Hebrew)
- top-to-bottom, then left-to-right (the normal reading sequence for Chinese)

As mentioned, there are potentially 8 choices of reading direction.

Contiguous arrays take cells confined to specified consecutive rows and consecutive columns. For example, we can specify a block of 6 cells occupying the 2nd and 3rd rows of the grid, and columns 3 through 5. To create more variety, it also permitted have **noncontiguous arrays** which skip a single row or column. For example, we might consider the array consisting of 3 rows and 5 columns located in the NW corner of the 6×6 grid. If we ignore the second row of this array, what remains amounts to an array with dimensions 2×5. We thus say that the **adjusted dimensions** of the 3×5 array after skipping the second row are 2×5. Note that no noncontiguous array is obtained by skipping the first or third row of the original 3×5 array; this would give only ordinary, contiguous 2×5 arrays. Also, by convention, we never skip both a row and a column.

An $m \times n$ contiguous array can be positioned in the 6×6 grid in exactly $(7-m)(7-n)$ ways. If $m < 6$, we can obtain a non-contiguous array of (adjusted) dimensions $m \times n$ by deleting one of the $m-1$ interior rows of an $(m+1) \times n$ contiguous array. The larger contiguous array can be chosen (i.e., positioned in the grid) in $(7-(m+1))(7-n)$ ways. Thus there are $(m-1)(6-m)(7-n)$ ways to obtain a non-contiguous $m \times n$ array from an $(m+1) \times n$ array[1] Similarly, if $n < 6$, we can obtain a non-contiguous array of dimensions $m \times n$ by deleting one of the $n-1$ interior rows of an $m \times (n+1)$ contiguous array. The total number of ways of obtaining an $m \times n$ array, either contiguous or non-contiguous, is thus

$$\begin{aligned}(7-m)(7-n) + (m-1)(6-m)(7-n) + (n-1)(7-m)(6-n) \\ = (mn+48)(m+n) - 7(m^2+n^2) - 13mn - 35. \qquad (3)\end{aligned}$$

[1]Note that we may comprehend the case $m = 6$ in this formula, which then evaluates to 0.

Array style PassRules may use unary or binary operators. If we use unary operators with an array of adjusted dimensions $m \times n$, there will always be 8 choices for reading direction. The number of cells will be mn, so that by (1) the number of possible sub-rule sequences is $7 + 42\binom{mn}{2}$. We find that the number of unary PassRules given by arrays with adjusted dimensions $m \times n$ is

$$8\left((mn+48)(m+n) - 7(m^2+n^2) - 13mn - 35\right)\left(7 + 42\binom{mn}{2}\right).$$

In the array style, binary operators are only used when one of the adjusted dimensions is 2 and the other dimension is 4 or greater. Further, if the adjusted array has 2 rows, the reading direction is first vertical, then horizontal, for example, top-to-bottom, then right-to-left. This would apparently give 4 choices; however, top-to-bottom and bottom-to-top are equivalent here, since the order of two cells for a binary operator doesn't matter. Therefore, there are really only 2 choices. If the adjusted array has two-columns, the reading direction is first horizontal, then vertical, for example, right-to-left, then top-to-bottom, and there are again 2 (distinct) choices. In the case of a grid with (adjusted) dimensions $2 \times m$, $m = 4, 5$ or 6, for a binary PassRule we choose a sequence of binary sub-rules of length m, so that by (3) and (2) the number of PassRules is

$$2\left((2n+48)(2+n) - 7(4+n^2) - 26n - 35\right)\left(5 + 20\binom{2n}{2}\right).$$

This will also be the number of binary sub-rules where the adjusted dimensions are $n \times 2$, $n = 4, 5$ or 6. The counts of the various array style PassRules are given in Figure 2.3.2. Note that number of PassRules for, e.g., 2×6 and 6×2 are the same, but the last line takes into account all array PassRules together.

2.3.3 Line Style

We first count possible cell sequences for line style, organizing our count based on starting cell and sequence length. Under a rotation and possibly a reflection, each cell sequence can be made to start at one of cells 1, 2, 3, 5, 6 or 9.

For this report we omit a detailed enumeration relating edge properties, starting square and length. However, we note that not every edge property

dimensions	operators	number of PassRules	entropy
2×2	unary	134,680	17.04
2×3	unary	336,336	18.36
2×4	unary	539,448	19.04
2×4	binary	64,410	15.97
2×5	binary	68,780	16.07
2×6	binary	23,850	14.54
total		2,335,008	21.15

Figure 6: Counting PassRules in array style

start	length n												
	4	5	6	7	8	9	10	11	12	13	14	15	16
1	3	3	3	7	11	11	11	11	11	11	11	11	11
5	30	30	34	38	38	38	38	38	38	38	38	38	38
9	20	36	42	42	42	42	42	42	42	42	40	40	38
1, 5, 9	53	69	79	87	91	91	91	91	91	91	89	89	87
2	12	12	16	21	22	22	22	22	22	22	22	22	21
3	9	17	21	22	24	24	24	23	23	23	23	23	23
6	28	37	42	44	44	43	42	42	42	41	41	39	39
2, 3, 6	49	66	78	87	90	89	88	87	87	86	86	84	83

Figure 7: Cell sequences in line style by start

can be used with every length. Also, for example, with respect to the SW direction, edge properties **left turn**, **u-turn-left-then-right** and **bounce II** all describe the edge property of cell sequence 6-8-19-22. We tabulate the number of possible line sequences of each length starting at cells 1,2,3,5,6 and 9 in Figure 3

Each of cells 1, 5 and 9 represents 4 cells from which the same number of line sequences start. For example, the four corners 1, 12, 25, 36 will each start the same number of cell sequences. Also cells 2, 3 and 6 each 'represent' 8 other starting points. It follows that the total number of line style cell sequences of each length is as given in Figure 4. Since in line style only unary operators are used, PassRules may be counted for each length by multiplying the number of cell sequences by (1).

length n												
4	5	6	7	8	9	10	11	12	13	14	15	16
604	804	948	1044	1084	1076	1068	1060	1060	1052	1042	1028	1012

Figure 8: Counting cell sequences in line style

length n	number of PassRules	entropy
4	156,436	17.26
5	343,308	18.40
6	603,876	19.20
7	928,116	19.82
8	1,282,372	20.29
9	1,634,444	20.65
10	2,025,996	20.95
11	2,456,020	21.22
12	2,945,740	21.50
13	3,453,716	21.73
14	3,997,476	21.93
15	4,450,676	22.12
16	5,107,564	22.29
all lengths	29,475,740	24.81

Figure 9: Counting PassRules in line style

2.3.4 Quadrant Style

In **quadrant style** we choose a sequence of distinct cells in the NW quadrant of the 6×6 grid. If unary operators are used we pick 1 or 2 cells; if binary operators are used, 2 are 4 cells are chosen. The cell sequence is then echoed in the other 3 quadrants of the grid via one of 4 variations: translation, left-right reflection, top-bottom reflection, or both left-right and top-bottom reflection. Note that if the cell sequence in the NW quadrant lies in the second column, translation and left-right reflection give the same result. If the sequence lies in the second row, then translation and top-bottom reflection are indistinguishable. After a cell sequence has been chosen in each of four quadrants, the quadrants are ordered in one of $4! = 24$ possible ways to give the entire sequence of cells.

We start with the case where a single cell is chosen in the NW quadrant. If we choose the central cell of the quadrant, then all reflections and translation give the same result. From such a choice then, exactly 24 cell sequences arise. If we choose instead one of the other cells in the second column, two different ways of echoing the cell are possible; translation and left-right reflection have the same effect; top-bottom reflection and top-bottom/left-right double reflection are also mutually indistinguishable. This choice then gives $2 \times 24 = 48$ cell sequences. A similar result holds if we choose one of the other two NW quadrant cells in the second row. If, however, we choose a cell in the corner of the quadrant, all four 'echoes' are distinct, and we get $4 \times 24 = 96$ cell sequences. The total number of cell sequences corresponding to a choice of one cell in the NW quadrant is thus

$$24 + 4 \times 48 + 4 \times 96 = 576.$$

Now consider two cells in the NW quadrant. A sequence of two cells in the second row of this quadrant can be chosen in 6 ways. In this case, translation is the same as top-bottom reflection, and right-left reflection is the same as double reflection. The situation thus gives right to

$$6 \times 2 \times 24 = 288$$

cell sequences. The same number of cell sequences arise when we choose a sequence of two cells in the second column of the quadrant. The final case occurs when we choose our two cell sequence in one of the $9 \times 8 - 2 \times 6 = 60$ other ways. In such cases, all four ways of 'echoing' of the sequence are

distinct, and

$$60 \times 4 \times 24 = 5,760$$

cell sequences result. The total number of cell sequences corresponding to choosing a two-cell sequence in the NW quadrant is thus

$$288 + 5760 = 6048.$$

If we wish to use two cells in the NW quadrant for a binary operation, then the order of the cells doesn't matter. This means that if we choose two cells in the NW quadrant in positions that are symmetrical about the second column of the NW quadrant, then translation and left-right reflection will create the same PassRule. Similarly, top-bottom reflection and double reflection have the same result. An analogous observation holds if the two cells are symmetric about the second row of the NW quadrant. Two pairs (2-8, 4-6) are symmetric about both these axes. For these pairs, all ways of echoing amount to the same thing. There are four other (1-3, 7-9, 2-5, 5-8) pairs which are symmetric about the second column. Similarly, 4 pairs are symmetric about the second row, but not the second column. Thus for two pairs, 24 sequences for binary quadrant PassRules arise; for eight pairs $2 \times 24 = 48$ sequences arise; for the remaining $\binom{9}{2} - 10 = 26$ pairs, $4 \times 24 = 96$ sequences arise. The total number of cell sequences distinguished by quadrant PassRules using binary operators, corresponding to choosing a two-cell pair in the NW quadrant is thus

$$2 \times 24 + 8 \times 48 + 26 \times 96 = 2,928.$$

We can choose 4 cells (arranged as a sequence of 2 pairs of cells) in the NW quadrant in $\binom{9}{2} \times \binom{7}{2} = 756$ ways. If both of our sequence pairs are taken from 3-6 and 2-8, then all reflections will have the same effect as simple translation. There are two such sequence pairs, each giving rise to only $4! = 24$ cell sequences. If we choose one pair from {2-8, 3-6}, and the other from among {1-3, 7-9, 2-5, 5-8, 1-7, 3-9, 4-5, 5-6}, then reflection along exactly one of the two horizontal and vertical axes will matter. We may form a sequence of two pairs of this sort in $2 \times 2 \times 8 = 32$ ways, and there will be $2 \times 24 = 48$ resulting cell sequences after translation/reflections. All of the other $756 - 34 = 722$ sequences of two pairs will give 4 distinct cell sequences after translation/reflection. We therefore find that the total number of sequences to be considered arising from 4 cells in the NW quadrant

quadrant cells	operators	number of PassRules	entropy
1	unary	149,184	17.19
2	unary	7,154,784	22.77
2	binary	366,000	18.48
4	binary	40,056,240	25.26
all types		47,726,208	25.51

Figure 10: Counting PassRules in quadrant style

is

$$2 \times 24 + 32 \times 48 + 722 \times 96 = 70,896.$$

Using (1) and (2), we see that the total numbers of quadrant style PassRules are given by

- unary rules on 4 cells: $576 \times 259 = 149,184$.
- unary rules on 8 cells: $6,048 \times 1,183 = 7,154,784$.
- binary rules on 8 cells: $2,928 \times 125 = 366,000$.
- binary rules on 16 cells: $70,896 \times 565 = 40,056,240$.

2.3.5 Pair Style

The **pair style** PassRules [4] use binary operators on cells given by sets of adjacent pairs. No cells are reused, and the adjacencies are either all horizontal or all vertical. To begin this analysis, let us first restrict our attention to the case of pairs which are adjacent horizontally. To count pair style PassRules, consider how m pairs are to be distributed in a 6×6 grid. We consider first the ways k horizontal pairs can be situated on a single line of the grid:

- There is only one way to place 3 non-overlapping pairs on a line.
- There are 6 ways to place 2 non-overlapping pairs on a line.
- There are 5 ways to place a single pair on a line.

Now consider the ways that a set of m pairs may be spread among the 6 grid lines. For example, in the case $m = 8$ we may place 3 pairs on each of two grid lines, the remaining 2 pairs on a third line; equally, we could have placed 2 pairs on each of four different grid lines. We are led to consider the ways of partitioning 8 into (at most 6) parts of size 3 or less:

$$\begin{aligned} 8 &= 3+3+2 = 3+3+1+1 = 3+2+2+1 = 3+2+1+1+1 \\ &= 3+1+1+1+1+1 = 2+2+2+2 = 2+2+2+1+1 \\ &= 2+2+1+1+1+1. \end{aligned}$$

If we choose, say, two lines to contain 3 pairs and two to contain 1 pair (corresponding to $8 = 3+3+1+1$), there are $\binom{6}{2}\binom{4}{2}$ ways to choose the relevant lines. The total number of ways of choosing the 8 pairs in this case is thus

$$\binom{6}{2}\binom{4}{2} \times 1 \times 1 \times 5 \times 5$$

since there is 1 way to fill a line with 3 pairs, but 5 ways to place 1 pair on a line.

Regardless of which 8 pairs are chosen, there are 8! ways to order the pairs. A sequence of 8 binary operators with no transitions may be chosen in 5 ways, since there are exactly 5 binary operators. To choose a sequence with one or two transitions, first choose where the transitions occur; this may be done in

$$\binom{7}{1} + \binom{7}{2} = \binom{8}{2}$$

ways. Then choose a first and a second binary operator, which can be done in $5 \times 4 = 20$ ways. For a given choice of 8 pairs, there are thus

$$8!\left(5 + 20\binom{8}{2}\right) \tag{4}$$

Number of Pairs	Number of Pair Style PassRules	Entropy
4	114,390,000	26.78
5	1,875,258,000	31.81
6	98,626,752,000	36.52
7	2,122,764,840,000	40.95
8	37,574,651,520,000	45.10
total	39,798,032,760,000	45.18

Figure 11: Counts and entropies for pair style PassRules with given numbers of pairs

PassRules. From our listing of partitions, we see that the number of ways of choosing 8 pairs is

$$\begin{array}{lll} & \binom{6}{2}\binom{4}{1} \times 1^2 \times 6 & \text{from } 8 = 3+3+2 \\ + & \binom{6}{2}\binom{4}{2} \times 1^2 \times 5^2 & \text{from } 8 = 3+3+1+1 \\ + & \binom{6}{1}\binom{5}{2}\binom{3}{1} \times 1 \times 6^2 \times 5 & \text{from } 8 = 3+2+2+1 \\ + & \binom{6}{1}\binom{5}{1}\binom{4}{3} \times 1 \times 6 \times 5^3 & \text{from } 8 = 3+2+1+1+1 \\ + & \binom{6}{1}\binom{5}{5} \times 1 \times 5^5 & \text{from } 8 = 3+1+1+1+1+1 \\ + & \binom{6}{4} \times 6^4 & \text{from } 8 = 2+2+2+2 \\ + & \binom{6}{3}\binom{3}{2} \times 6^3 \times 5^2 & \text{from } 8 = 2+2+2+1+1 \\ + & \binom{6}{2}\binom{4}{4} \times 6^2 \times 5^4 & \text{from } 8 = 2+2+1+1+1+1 \end{array} \tag{5}$$

Multiplying (4) by (5), we find that the total number PassRules with 8 horizontal pairs is thus $18,787,325,760,000$, with an entropy of 44.09. It follows that the total number of PassRules in pair style with 8 pairs has entropy $44.09 + 1 = 45.09$.

Clearly, entropy can be very large even restricting to wizard-generated PassRules. For completeness, we list the number of PassRules and their corresponding entropies for the various numbers of pairs in Figure 7.

2.3.6 Summary of Wizard Styles

As mentioned, the various wizard-generated styles are not disjoint. We can slightly overestimate the number of all wizard-generated PassRules by simply

adding the total counts for all four styles. This gives an entropy result of

$$\log_2 39,798,111,129,452 \approx 45.18$$

One sees that this is identical (at least, to 2 decimal places) with the total entropy contribution of the pair style PassRules, which is in turn only very slightly higher than the contribution of pair styles PassRules on 8 pairs. The total contribution to entropy together of array, line and quadrant styles is overestimated by

$$\log_2 78,369,452 \approx 26.22$$

which is comparable to that of (all rules) of the trivial protocol on a 5×5 grid. In this entropy, one sees that almost none of the contribution from these three styles comes from array style.

3 PIN guessing

In [9], Weber makes the interesting observation that in the trivial 5×5 scheme the odds of guessing the PIN corresponding to a random grid and a hidden cell sequence are improved by guessing a random cell sequence and reading the corresponding PIN from the grid, rather than directly guessing a PIN. We examine (and correct) Weber's results, and consider a few analogous results related to PassRules™.

3.1 PIN guessing in uniformly random grids

We will begin with the case where grids are filled uniformly at random; that is, for each grid cell, independently, a digit from 0 to 9 is chosen uniformly and randomly. Weber gives a formula for the probability that a randomly chosen cell sequence gives a PIN which matches that given by a user's secret cell sequence. In fact, Weber's formula contains an error, but the overall guessing strategy is sound. If two 4-cell cell sequences are chosen for an $m \times m$ grid, $m \geq 4$, and the cells of the grid are filled uniformly and randomly, the true probability of the cell sequences giving the same PIN is actually

$$\Big(m^{14} + 36\,m^{12} + 594\,m^{10} + 5040\,m^8 + 17649\,m^6$$

$$-4356\,m^4 - 16884\,m^2 + 7920\Big)\,/10^4 m^{14} \tag{6}$$

For various m, the probabilities $P(m)$ of correctly guessing a PIN by guessing a cell sequence round up to

m	$P(m)$
4	0.00070650
5	0.00037576
6	0.0002577
7	0.00020280
8	0.00017278

Weber's error is in an unfortunate direction, in that he overestimates the security of the trivial 5×5 scheme by giving $P(5) = 0.000342102$. In fact in the scheme studied, one is almost 4 times as likely to guess a PIN by choosing a cell sequence as one is to simply guess a randomly selected 4-digit PIN.

Recognizing that mistakes may be made in ambitious probability formulas such as (6), one can alternatively run simulations as follows:

1. Randomly fill an $m \times m$ grid.
2. Pick two random 4-tuples of grid positions.
3. See if the PINs match.
4. Repeat many times.

The proportion of matching PINs estimates the probability $P(m)$. The number of times one has to repeat these steps to get a reliable estimate depends on the desired accuracy, and also on the true probability $P(m)$. If one accepts that believes that the true probability $P(m)$ is less than 0.0004 for $m = 5, 6, 7, 8$, one can estimate $P(m)$ 98% of the time to within 0.0001 by running the simulation steps 217,070 times. A run of the stated simulation 217,070 times for $m = 5, 6, 7, 8$ gave these values:

m	$P(m)$
5	0.00036854
6	0.00027180
7	0.00021652
8	0.00017278

as expected, given the exact calculations. It is clear that when grids are filled uniformly and randomly and no operators are used, one can guess PINs corresponding to grids and no operators somewhat more easily via guessing cell sequences.

3.2 PIN guessing in balanced grids

A solution proposed by Weber to the weakness noticed in the last section is **grid balancing**. Instead of filling grid cells independently and randomly, one chooses ways of filling the cells such that the numbers of each digit in the grid are as equal as possible. It is again possible to calculate the probabilities that PINs corresponding to random 4-tuples of cells match in such a scenario. An exact formula could be obtained with much effort, but, as noted in the last section, the probabilities can be estimated as closely as desired by simulation.

From the point of view of PassRules™, the most important case occurs when $m = 6$. Suppose that two random 4-tuples of cells from a 6×6 grid are chosen, and the grid is randomly filled in such a way that 4 of the digits from $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ appear 3 times among the grid cells, while the other 6 appear 4 times, with every such way of filling the grid equally likely to be chosen. Running such a simulation gave a probability value 0.00010135 that the 4-tuples give the same passcode. This is essentially the same as guessing a 4-digit PIN with no additional knowledge, i.e. 0.0001.

3.3 PIN guessing in other tailored grids

We have seen that if a PassRule gives a 4-digit passcode, we always have a probability of at least 0.0001 of guessing the PIN, simply by guessing 4 digits at random as a PIN. As noted, if a challenge grid is filled with random digits, this probability can be improved quite a bit by picking a random PassRule, and then reading a passcode from the grid. However, if the grid is chosen to be **balanced** – if cells are filled randomly, but in such a way that each digit occurs in exactly 3 or 4 cells – the probability of guessing a passcode via a random PassRule is again reduced to approximately 0.0001.

It should not be thought that balanced grids are optimal from the point of view of reducing the probability of guessing a PIN via a random PassRule. For example, consider a trivial PassRule which simply echoes the contents

of cells 1, 2, 3 and 10 in a grid. (In other words, all 4 operators are +0.) Consider the following grid:

X:

7	8	9	9	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

The correct passcode corresponding to this grid is 7-8-9-9. Only 4 trivial PassRules give this passcode. (We pick from two cells to get the 9, and repetition is allowed.) The total number of trivial length 4 PassRules is $36^4 = 1,679,616$, so that the probability of guessing a trivial PassRule yielding 7-8-9-9 from this grid is

$$\frac{4}{1,679,616} \approx 0.000002381$$

This is only about $\frac{1}{42}$ the probability of guessing a 4-digit PIN! Here, we are assuming that we know that we are dealing with a trivial length 4 PassRule. What if we know only that a length 4 PassRule is involved?

The number of possible PassRules of length 4 is 2,053,886,447,390,976. There are only a few ways that 7 can be returned from the given grid via a sub-rule: $7 = 7 + 0$ (unary rule), $7 = 7 + 0$ (binary rule), $7 = 7 - 0$, $7 = 7b0$, $7 = 7b7$, $7s7$, $7s8$, $7s9$. In the case $7 = 7 + 0$ (unary rule), there is only one cell choice yielding 7. In the case $7 = 7 + 0$ (binary rule) there is a single cell giving 7, and 32 cells yielding 0. Thus the complete sub-rule, including choice of cells, may be specified in 32 ways. The same is true for $7 - 0$ and $7b0$, while $7b7$, $7s7$, $7s8$ each correspond to a unique choice of cells. The rule $7s9$ may be applied with exactly 2 cell choices. We see then that the sub-rule generating the 7 of the passcode may be chosen in exactly 102 ways. The same is true of the sub-rules generating 8. There are 105 sub-rules giving 9. Not every set of 4 sub-rules giving 7-8-9-9 will be memorable. Thus the probability that a randomly chosen memorable sub-rule applied to the given grid yields 7-8-9-9 is strictly less than

$$\frac{102 \times 102 \times 105 \times 105}{2,053,886,447,390,976} \approx 0.00000005585$$

This is only about $\frac{1}{1791}$ the probability of guessing a 4-digit PIN! We see that by tailoring a grid to match a PassRule, the probability that a random PassRule gives a PIN matching a hidden PassRule can be made extremely small. The difficulty is that an adversary is not restricted to finding PINs via guessing PassRules. In fact, the values 7, 8, 9, 9 stand out like a sore thumb in the given grid, and a human adversary seeing the grid would likely guess one of the $4!/2 = 12$ PINs consisting of a permutation of 7-8-9-9, giving a

$$\frac{1}{12} \approx 0.0833$$

probability of guessing our PIN. (This is another example of entropy leakage in tailored grids.)

4 Rule capturing

4.1 Phishing

It is now a daily occurrence to receive spurious messages in one's email accounts, purporting to come from various financial institutions. Generally users are urged to login to accounts via a link in order to correct or avoid some security breach. In reality the link will be to a web page which is a counterfeit of the legitimate login page of the financial institution; account and password information entered will go to criminals. This sort of large-scale social engineering attack is characterized as **phishing**. Evidently, any password scheme whatsoever is vulnerable to phishing attacks, which in fact attack, not the scheme itself, but human weakness. Any user who enters an account number and a password has become a victim, even if the password is a 40 digit random string of numbers and letters.

4.1.1 Phishing versus trivial operators

As Bond points out grid based schemes can also be also vulnerable to phishing attacks. In the 5×5 case with no operators he gives the example of these grids:

A:

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

B:

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

If a user fills in 4 digit passcodes for both these two grids, he has entirely specified his cell sequence. Of course these particular grids would hopefully arouse suspicion. Bond mentions the possibility of disguising these grids by permuting rows and columns. Much more is possible; from a mathematical point of view, what is going on is that the 25 pairs (a_i, b_i), where a_i is in cell i of grid A and b_i is in cell i of grid B, are all distinct. Since there are $10 \times 10 = 100$ possible choices for ordered pairs of digits, and only 25 grid positions to fill, there are $25!\binom{100}{25} \approx 10^{48}$ pairs of grids, like Bond's, which could expose the secret if used in a phishing attack. For example, the random-appearing grids

1	4	9	1	1
2	6	7	2	9
9	3	6	0	3
8	4	6	4	4
4	1	3	5	0

1	2	3	2	3
1	1	7	4	5
1	9	2	4	6
1	4	9	6	8
0	4	3	0	5

would work. Since the user must enter a passcode twice, the trivial scheme using a 5×5 grid and no operators is safer than the normal PIN scheme with respect to phishing, but not by much.

PassRules™ is also vulnerable to phishing attacks. In the trivial non-altering version of PassRules™, a 6×6 grid is used and no obscuring operators are used. However, simply using a 6×6 version of Bond's grids A and B exposes the secret in this case. Note that the security is not increased here by increasing the length of the PassRule/passcode.

4.1.2 Phishing versus unary operators

One step up from the base level of PassRules™, one might use only the unary obscuring operators $\{+0, +1, \ldots, +6\}$. Surprisingly, this scheme is also very

vulnerable to phishing. Any PassRule™ using only unary operators can be determined from responses to these three grids:

C:

1	4	9	1	1	5
2	6	7	2	9	6
9	3	6	0	3	7
8	4	6	4	4	8
4	1	3	5	0	9
7	6	5	4	0	2

D:

1	2	3	2	3	8
1	1	7	4	5	0
1	9	2	4	6	8
1	4	9	6	8	6
0	4	3	0	5	2
1	6	3	7	6	9

E:

2	4	6	4	6	6
2	2	4	8	0	0
2	8	4	8	2	6
2	8	8	2	6	2
0	8	6	0	0	4
2	2	6	4	2	8

Given the response to these three grids, any unary PassRule may be reconstructed in its entirety. Standard practice in password based systems is to cut users off after 3 failed password attempts. It would therefore be feasible for a phishing site to ask for a passcode up to three times with different grids. We conclude that phishing is a potential vulnerability for unary PassRules™, although slightly less so than for PINs or for the GridIDsure scheme. The same vulnerability could be attacked by Trojan horse password stealing programs. Again, even robust passwords fail in the face of such social engineering attacks, so this is not a special weakness of PassRules™.

How does phishing work here? In the above example, every entry of grid E is twice the corresponding entry in grid D, chopping values 10 or more. Suppose the unary subrule reads a value a in grid D, and adds constant value u. The result from grid D is some value $d = a + u$, while the result from grid E is some value $e = 2a + u$. The resulting system

$$\begin{aligned} d &= a + u \\ e &= 2a + u \end{aligned}$$

can be solved by linear algebra to yield a and u. Since we now know the obscuring operator is $+u$, we can deduce the cell value read by the sub-rule in grid C, D and E. The two grids C and D are constructed in such a way that knowledge of a cell's contents in both grids pins down the cell location, just like in Bond's grids A and B. Therefore, knowing a sub-rule result from all 3 grids specifies the sub-rule completely. Although chopping is used to form grid E, whether our sub-rule protocol is natural, or involves chopping or padding, responses to these grids will decode any unary PassRule.

The grids C and D here were specifically chosen to appear random, in order to illustrate that a phishing attack need not be obvious. A more trans-

parent version of this attack would be given by:

$\mathcal{C}$:

0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5

$\mathcal{D}$:

0	1	2	3	4	5
0	1	2	3	4	5
0	1	2	3	4	5
0	1	2	3	4	5
0	1	2	3	4	5
0	1	2	3	4	5

$\mathcal{E}$:

0	2	4	0	2	4
0	2	4	0	2	4
0	2	4	0	2	4
0	2	4	0	2	4
0	2	4	0	2	4
0	2	4	0	2	4

which works in exactly the same way, but involves highly suspicious-looking grids. In view of the interest of balanced grids, one might ask whether a version of this construction can be given where all three grids are balanced. The answer is yes:

Say that two grids are **complementary** if knowing a cell's contents in the two grids pins down the cell's location. Given any balanced grid $\mathcal{A}$, it is easy to write down a complementary balanced grid $\mathcal{B}$ as follows: In PassRules™, we have a numbering of grid cells from 1 to 36. In grid $\mathcal{A}$, find the cell containing a 0 (there will be either 3 or 4 of them) with the lowest grid number. Place a 0 in the corresponding cell of $\mathcal{B}$. Now find the cell in $\mathcal{A}$ containing a 0 with the second lowest grid number and place a 1 in the corresponding cell of $\mathcal{B}$. In the cell of $\mathcal{B}$ corresponding to the third lowest 0 of $\mathcal{A}$ place a 2. If there is a fourth 0 in $\mathcal{A}$, a 3 goes in the corresponding cell of $\mathcal{B}$; otherwise we start with the lowest 1 of $\mathcal{A}$. Continuing in this way we fill the cells of $\mathcal{B}$ with $0, 1, 2, \ldots, 9, 0, 1, 2, \ldots$, producing a balanced filling of $\mathcal{B}$. The cells containing 0 (or any other given value) in $\mathcal{A}$ must all receive different values in $\mathcal{B}$, so the two grids are complementary. The two grids below illustrate the construction of $\mathcal{B}$ partially finished:

$\mathcal{A}$:

0	1	2	3	4	5
1	0	9	8	7	6
7	2	3	4	5	6
9	0	8	1	3	2
3	4	6	5	8	7
9	4	3	2	1	0

$\mathcal{B}$:

0	4	8			
5	1				
	9				
	2		6		0
				7	3

If we create complementary grids $\mathcal{A}$ and $\mathcal{B}$, then adding a third grid equal to $2\mathcal{A}$ will give a phishing scheme that defeats any unary PassRule.

However, it will be correctly objected that $2\mathcal{A}$ contains all even numbers, and therefore is not balanced. This flaw may be fixed by insisting that instead of $2\mathcal{A}$ we form a grid where 0, 1, 2, 3, 4 from $\mathcal{A}$ are doubled, but 5, 6, 7, 8, 9 are doubled, and then increased by 1 and chopped. Such a grid must be balanced. In the present case, grid $\mathcal{F}$ results:

$\mathcal{F}$:

0	2	4	6	8	1
2	0	9	7	5	3
5	4	6	8	1	3
9	0	7	2	6	4
6	8	3	1	7	5
9	8	6	4	2	0

This is just as good as having $2\mathcal{A}$; if a sub-rule gives an odd result, we simply subtract 1 to find $2a + u$ as before, and solve by linear algebra.

It may be objected that it is unrealistic to launch a phishing expedition against specifically unary rules. How would an adversary know that a rule is unary? Suppose, one would assume will generally be the case, that wizard-generated rules are in use. An adversary does not know whether a unary or binary rule is in effect. In fact, the great majority of the wizard-generated PassRules are binary rules given by pair style, so that guessing that a rule is binary would make more sense. However, in some cases it is crystal clear that a unary rule is being used– only unary rules will use PassRules of lengths 9 through 16. If a PassRule is seen to have one of these lengths, then it must be unary. We are here assuming that chopping, padding or grid tailoring is used, so that PassRule lengths are always known (Rule length may also be guessed via timing attacks.) However, even if natural sub-rule results are used, a PassRule which returns a code of length $17 - -32$ must be unary.

In our analysis of entropy, we distinguish between PassRules of lengths 4 and 5. For phishing, the length of a unary rule is irrelevant. Since line style PassRules are unary, the above analysis shows how phishing can be used to attack any line style PassRule. We now turn our attention to binary rules.

4.1.3 Phishing against binary rules

As we have observed, the bulk of the wizard-generated PassRules are pair style rules. In considering phishing and binary PassRules, the most impor-

tant topic is thus perhaps whether pair style rules are vulnerable to phishing. Consider the following balanced grid:

$\mathcal{G}$:

0	1	2	3	4	3
5	6	7	8	9	8
0	1	2	3	4	3
5	6	7	8	9	8
0	1	2	3	4	3
5	6	7	8	9	8

Recall that pair style operates on pairs of adjacent cells. In this grid, possible results returned by binary operators on adjacent cells are

$+$	1, 3, 5, 7,9
$\times$	0, 2, 4, 6
$-$	1, 5
s	0,1,2,3,4,5,6,7,8
b	1,2,3,4,5,6,7,8,9

Now consider the balanced grid $\mathcal{H}$, where an entry x in $\mathcal{G}$ is replaced by $x + 1 \pmod{10}$[2]:

$\mathcal{H}$:

1	2	3	4	5	4
6	7	8	9	0	9
1	2	3	4	5	4
6	7	8	9	0	9
1	2	3	4	5	4
6	7	8	9	0	9

An adjacent pair of cells (x, y) in grid $\mathcal{G}$ resulting in $x + y = 5$ under addition in grid $\mathcal{G}$ would yield $(x + 1) + (y + 1) = x + y + 2 = 7$ in grid $\mathcal{H}$. More generally, the results of $+$ are shifted according to the cycle $1 \to 3 \to 5 \to 7 \to 9 \to 1$. The results of $\times$ remain even numbers; in particular, a product 4 from grid $\mathcal{G}$ shifts as either $2 \times 7 \to 3 \times 8 \equiv 4$ or $3 \times 8 \to 4 \times 9 \equiv 6$

[2]In other words, results greater than 10 are 'chopped'.

in grid $\mathcal{H}$. This is different from the case with b and s which are increased by exactly 1 moving from $\mathcal{G}$ to $\mathcal{H}$, except in the case where the adjacent pair in $\mathcal{G}$ was $(4, 9)$ or $(8, 9)$, in which case the results are shifted as $4 \to 0$, $8 \to 0$ under s, and $9 \to 5$, $9 \to 9$ under b. We conclude that we can distinguish, via these two grids, between $+$, $\times$ and b/s. In $\mathcal{H}$, possible results of $-$ are $1, 5, 9$, and the two grids thus also distinguish $-$ from all the other operators.

It follows that only two grids are required to recover a pair style operator sequence, up to a possible confusion of s and b. This represents a substantial loss of entropy. Consider, for example, the case where a pair style rule applied to $\mathcal{G}$ gives 1-2-0-2-1-5-7-3, and applied to $\mathcal{H}$ gives 3-0-2-0-3-7-9-5. Before reading the given passcodes, we had a pair style PassRule with 8 pairs, and thus an entropy of 45.10. Consider the last digits of the two passcodes. We see the shift $3 \to 5$, indicating that the last sub-rule uses $+$. Reasoning in a similar way, we find that the operator sequence in fact must be $+ \times \times \times + + ++$. Returning to the last digits, there are 6 possible horizontal additions in grid $\mathcal{G}$ giving 3 (namely $1 + 2$ and $6 + 7$ possible in various rows) and 5 vertical ones (namely $4 + 9$ in column 5). Note that all pairs in a pair style rule have the same orientation, horizontal or vertical, and cells are not reused. Therefore, if we generate the final 3 from the passcode corresponding to grid $\mathcal{G}$ using $1+2$ in the first row, it is no longer possible to use $2+3$ in the first row to generate the 5 of the passcode. In fact, each way of generating a 3 using a horizontal addition eliminates one way of generating a horizontal 1, and one way of generating a horizontal 5.

Suppose that 1-2-0-2-1-5-7-3 was indeed generated by horizontal pairs. There are 6 ways the final 3 could have been generated. This leaves 5 ways to choose the first 1, and then 4 ways to choose the second 1. There are similarly 5 ways remaining to choose a horizontal pair summing to 5. The choice of a horizontal pair adding to 5 eliminates the possibility of one of the horizontal pairs summing to 7 being used; 11 choices remain. Now we notice that a horizontally adjacent pairs has product 2 if and only if it has sum 3 or 7. One of the six 3's has been used, at least 2 more are eliminated by the two 1's. A 7 has been used, and one is eliminated by the 5. It follows that the first pair with product 2 can be chosen in at most $18 - 3 - 2 = 13$ ways, the second in 12 ways. Horizontal pairs have product 0 if and only if they sum to 1. There are thus at most 3 choices remaining to produce product 0. The total number of ways to produce the observed passcode using disjoint

horizontal pairs is thus at most

$$6 \times 5 \times 4 \times 11 \times 13 \times 12 \times 3 = 3,088,800.$$

To count the possibilities with vertical pairs, consider the third digits of both passcodes. The only vertical pairs in $\mathcal{G}$ giving product 0 are in the first column. However, pairs in the first column of $\mathcal{H}$ give product 6, not 2 as required. No PassRule with vertical pairs will work! It follows that the entropy of the hidden PassRule has shrunk to

$$\log_2 3,088,800 \approx 21.56$$

4.2 Rule capturing via random grid and response

We have seen in the last section that if an adversary is able to send tailored grids to a user, at least some variations of PassRules™ can be broken. What if an adversary cannot send grids of his own choosing, but does have access to a user's responses to several grids? This could be the case, for example, if a secret camera is set up near an ATM.

Bond [2] brings up the issue of **entropy leakage** when a grid and its response are known. His analysis is only preliminary, as will be explained, but the principle is sound. Indeed, PassRules™ and similar schemes will always have to balance protection against the PIN guessing attacks of the previous section with protection against entropy leakage.

When a grid and its response are known, all PassRules which would not give the known response to the given grid are ruled out. Suppose that in some particular protocol a secret PassRule s initially has entropy E; that is, there are 2^E PassRules possible. Now let a given grid be given, and suppose that the probability that a random PassRule would give the same passcode as s is p. This means that the number of PassRules giving the same passcode is $p2^E$. It follows that if the grid and the passcode associated with s are known, $(1-p)2^E$ PassRules are immediately ruled out as possibilities for s. The new pool of possible PassRules s has entropy $\log_2\left(p2^E\right) = E + \log_2 p$. Since p is a probability, $p \leq 1$, and $\log_2 p \leq 0$. The smaller p is, the more entropy is lost every time a grid/response pair becomes known. If we insist that $p \approx 0.0001$ then one would expect that, as Bond claims, the entropy lost given a grid/response pair is $-log_2 0.0001 \approx 13.3$.

A straight-forward application of this observation would lead to the assumption that the entropy of s is reduced to 0 (i.e., s is exactly specified)

after observing about $\frac{E}{13.3}$ grid/response pairs. On this basis, Bond claims that

> Clearly after two challenges, up to 26.6 bits of information have been revealed in the response, yet the maximal entropy of a pattern is 18.2 bits. Given the typical entropy of a pattern will be much lower, due some common patterns being much more appealing than others, it is likely that knowledge of both challenge grid and response from a single challenge will yield enough information to determine the pattern fully in a significant proportion of cases, and two challenge/response pairs will suffice in most cases.

This critique is aimed at a trivial 5×5 scheme with no operators[3]. Evidently, the much greater entropies associated with PassRules™ could protect against this problem. Bond's analysis here is a first approximation. The phrase 'up to 26.6 bits of information have been revealed' could be reworded to say that 'no more than 26.6 bits of information can have been revealed'. In extreme cases, a grid/response pair may give no new information at all about a cell sequence. For example, there is a non-zero probability that subsequently generated grids are identical. While the chance of this extreme case is negligible, it illustrates that information leaked by a sequence of grid/response pairs will almost never be strictly additive; this is another way of saying that probabilities of guessing a correct passcodes on subsequent grids are not independent.

4.2.1 Entropy leakage estimates

Finding out how much entropy is lost in a fixed PassRules protocol when a grid/response pair is revealed is equivalent to measuring the probability p_1 that two random PassRules in the protocol give the same passcode in response to a random grid. As we have indicated, the expected loss of entropy is $-\log_2 p_1$.

As we stated in our section on PIN guessing, probability p_1 may be found to any given level of precision via simulation. Bond's preliminary analysis of entropy leakage assumes that the expected entropy loss when two grid/response pairs have been revealed is twice this amount, namely

[3]Bond restricts to no cells repeated, giving $\log_2(25 \times 24 \times 23 \times 22) \approx 18.2$

$-2\log_2 p_1$. Let p_2 be the probability that two random PassRules conforming to a style or protocol give the same passcode in response to both of two random grids. Rephrasing, Bond's assumption is that $p_2 = p_1^2$. However, we can also approximate p_2 by simulation. The quantity $-2\log_2 p_1 + \log_2 p_2$ represents any 'slippage' in Bond's estimate. We can find p_2 as follows:

1. Randomly fill two $m \times m$ grids in a balanced way.
2. Pick two random PassRules in a given class (pair style, unary, etc.)
3. See if the PassRules give matching passcodes for both grids.
4. Repeat many times.

The ratio of matches to number of trials approximates p_2. We calculate p_1 in a similar way, as explained earlier.

In Figures 12 and 13, for the trivial (no operator) PassRules protocols we give simulated probabilities p_1 and p_2, and the corresponding entropies remaining, after one and two grid/response pairs are observed. For example, in the case where we use the 'natural' sub-rule protocol and a balanced grid, the proportion of matches in 10,000,000 random trivial length 4 PassRules is

$$0.000106$$

The number of such PassRules possible has entropy

$$\log_2 36^4 \approx 20.68$$

After a random grid/response pair, we expect

$$36^4 p_1 \approx 178$$

PassRules to be consistent with the observed grid/response, so that the new entropy is about

$$\log_2 178 \approx 7.47$$

We see that for the first grid/response observation, the loss of entropy for the balanced grids is close to the predicted amount of 13.3. However, the loss caused by the second observation is slightly smaller than predicted. Note that, since the results here were obtained by simulation, the accuracy of our estimation of p_1 and p_2, and hence entropy loss, is not perfect. More

length	sub-rule protocol	balanced grid	p_1	p_2
4	natural	yes	0.00011	0.0000010
4	natural	no	0.00026	0.0000023
4	padding	yes	0.00011	0.0000017
4	padding	no	0.00026	0.0000019
4	chopping	yes	0.00011	0.0000011
4	chopping	no	0.00026	0.0000019
5	chopping	yes	0.000011	0
5	chopping	no	0.000033	0

Figure 12: Simulated probabilities for PassRules without operators (10,000,000 iterations)

elaborate simulations and variance reduction techniques could conceivably lead to sharper values for p_1 and p_2. Evidently, the values of p_2 for $n = 5$ cannot be exactly 0, since sometimes PassRules give matching codes (if, for example, we choose the same rule twice.) The entropy estimate of 0 after two observations, however, is reasonable, since the estimate for entropy lost in one grid/response pair is $5 \times \log_2 10 \approx 16.5$, and the initial entropy is only 25.85. For trivial PassRules protocols, Bond's estimates of entropy loss, while not exact, definitely prove consistent with with the observed data.

The same simulations/calculations were undertaken for memorable unary PassRules, and for pair style PassRules. (See Figures 14–17.) Perhaps because these rules are more structured, we see much greater entropy leakage than in the trivial protocols. For pair style rules, the leakage gets worse as we use more pairs, which is somewhat counter-intuitive. On the bright side, remembering the leakage/PIN guessing tradeoff discussed earlier, passcodes will be much harder to guess for these rules.

length	sub-rule protocol	balanced grid	initial entropy	1 grid/reponse pair observed	2 grid/response pairs observed
4	natural	yes	20.68	7.47	1.33
4	natural	no	20.68	8.77	1.95
4	padding	yes	20.68	7.52	1.51
4	padding	no	20.68	7.50	1.67
4	chopping	yes	20.68	7.50	0.89
4	chopping	no	20.68	8.71	1.60
5	chopping	yes	25.85	9.32	0
5	chopping	no	25.85	10.96	0

Figure 13: Approximate expected entropies for PassRules without operators (10,000,000 iterations)

length	balanced grid	p_1	p_2
4	yes	0.000034	0.00000001
4	no	0.000041	0
5	yes	0.0000029	0
5	no	0.0000034	0

Figure 14: Simulated probabilities for memorable unary PassRules

length	balanced grid	initial entropy	1 grid/reponse pair observed	2 grid/response pairs observed
4	yes	28.45	13.60	1.87
4	no	28.45	13.84	0
5	yes	34.17	16.00	0
5	no	34.17	15.77	0

Figure 15: Approximate expected entropies for memorable unary PassRules

length	balanced grid	p_1	p_2
4	yes	0.000131	0.00000015
4	no	0.000131	0.00000015
5	yes	0.0000142	0
5	no	0.0000143	0
6	yes	0.00000173	0
6	no	0.00000172	0
7	yes	0.000000202	0
7	no	0.000000202	0
8	yes	0.000000044	0
8	no	0.000000046	0

Figure 16: Simulated probabilities for pair style PassRules (50,000,000 iterations)

length	initial entropy	1 grid/reponse pair observed	2 grid/response pairs observed
4	26.78	13.87	4.14
5	31.81	14.71	0
6	36.52	17.38	0
7	40.95	18.71	0
8	45.10	20.72	0

Figure 17: Approximate expected entropies for pair style PassRules (50,000,000 iterations)

References

[1] R. Anderson, M. Bond & S. Murdoch. Chip and Spin, *Computer Security Journal* **22(2)** (2006): 1-6, http:// www.cl.cam.ac.uk/ ∼ sjm217/ papers/ cl05chipandspin.pdf.

[2] M. Bond. Comments on Gridsure Authentication, http:// www.cl.cam.ac.uk/ ∼mkb23 /research/GridsureComments.pdf. Accessed on June 1, 2010.

[3] N. Goertzen. Comparing PassRules with passwords, Internal report, *It'sMeSecurity*.

[4] N. Goertzen, PassRules Wizard Specifications, Internal report, *It'sMeSecurity*.

[5] N. Goertzen. PassRules Syntax, Internal report, *It'sMeSecurity*.

[6] Bond, M. & P. Zielínski. Decimalisation table attacks for PIN cracking, Technical Report UCAM-CL-TR-560, University of Cambridge Computer Laboratory, Cambridge UK, February 2003.

[7] J. Katz. Efficient Cryptographic Protocols Preventing 'Man-in-the-Middle' Attacks. PhD Thesis, Columbia University, 2002.

[8] M. Mannan, P.C. van Oorschot. Reducing threats from flawed security APIs: The banking PIN case, *Computers & Security* **28(6)** (2009).

[9] R. Weber, The Statistical Security of GrIDsure, http://www.gridsure.com/uploads/Stats%20report %20 – %20Richard %20Weber.pdf, University of Cambridge, Tech. Rep., June 2006.

[10] R. Johnson & G. Bharracharyya, Statistics: Principles and Methods, Wiley (2006).