Parallel Processing Letters

CELLULAR AUTOMATA ENCRYPTION
the reverse algorithm, *Z*-parameter and chain-rules

ANDREW WUENSCHE*
Discrete Dynamics Lab. www.ddlab.org
and University of Sussex, United Kingdom.

Received (received date)
Revised (revised date)
Communicated by (Name of Editor)

ABSTRACT

Chain-rules are maximally chaotic cellular automata (CA) rules that can be constructed at random to provide a huge number of encryption keys – where the CA is run backwards to encrypt, forwards to decrypt. The methods are based on the 1D CA reverse algorithm for directly finding pre-images, and the resulting *Z*-parameter, and rely on the essential property that chain-rules have minimal in-degree in their basins of attraction, usually an in-degree of just one for larger systems.

Keywords: encryption, cellular automata, chain-rules, chaos, basins of attraction, pre-images, in-degree, reverse algorithm, *Z*-parameter, DDLab

1. The CA reverse algorithm and basins of attraction

In the simplest cellular automata [3], each cell in a ring of cells updates its value (0,1) as a function of the values in its neighbourhood, size k. All cells update synchronously – in parallel, in discrete time-steps, moving through a deterministic forward trajectory, as illustrated in Figs. 1 and 2. Each "state" of the ring, a bit string, has one successor, but may have multiple or zero predecessors (pre-images).

A book, "The Global Dynamics of Cellular Automata" [4] published in 1992 introduced a reverse algorithm for finding the pre-images of states for any finite 1D binary CA with periodic boundary conditions, which made it possible to reveal the precise topology of "basins of attraction" – represented by state transition graphs – states linked into trees rooted on attractor cycles, which could be drawn automatically as in Fig. 3. The software was attached inside the back cover on a floppy disk – the origin of what later became DDLab [10].

As state-space necessarily includes every possible piece of information encoded within the size of its string, including Shakespeare's sonnets, copies of the Mona Lisa,

*andy@ddlab.org

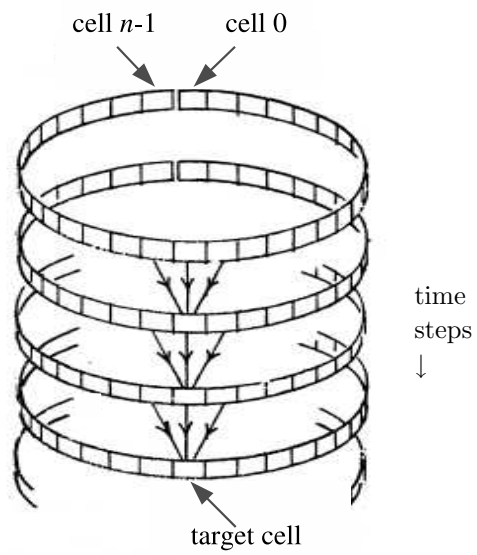

Fig. 1. Schematic of a 1D CA with periodic boundaries. Each cell updates in parallel as a function of the values in its neighbourhood at the previous time-step.

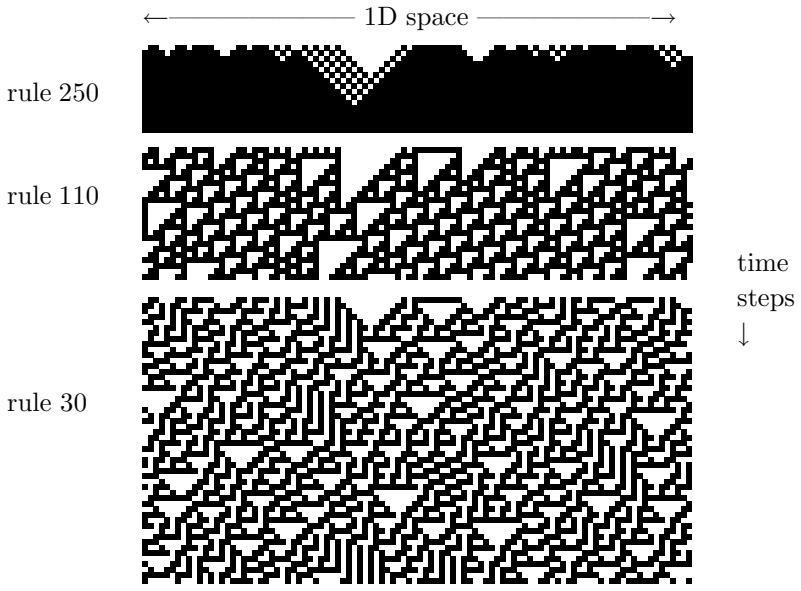

Fig. 2. 1D space-time patterns of the k=3 rules 250, 110 and 30, characteristic of order, complexity and chaos. Rule 30 is in fact an example of a maximally chaotic (chain) rule in k=3 rule-space. System size n=100 with (unfolded) periodic boundaries. The same random initial state was used in each case.

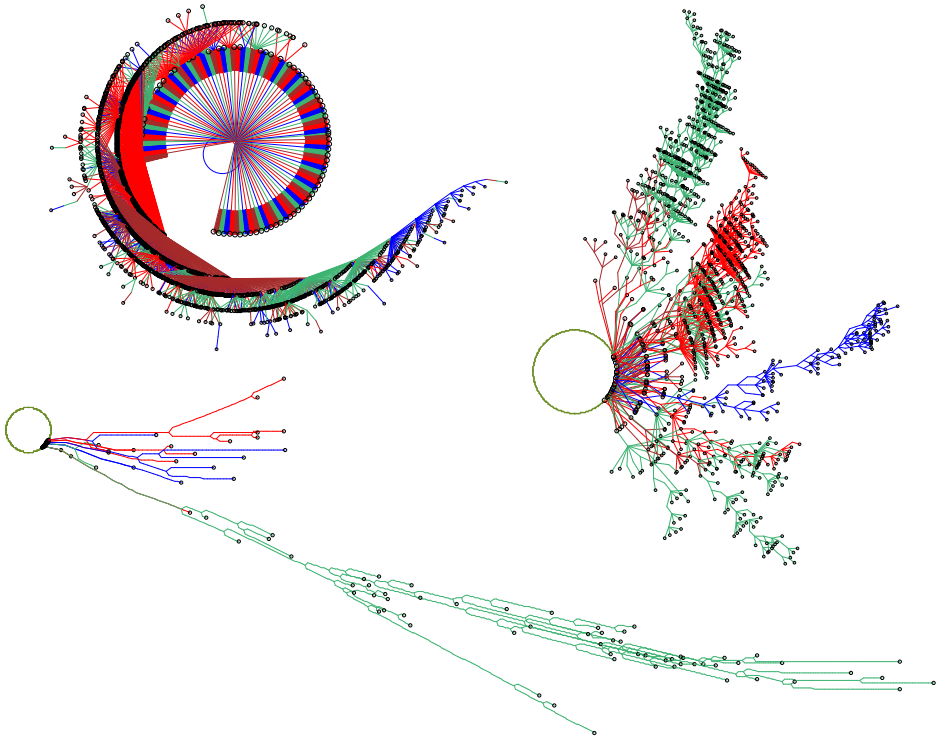

Fig. 3. Three basins of attraction with contrasting topology, n=15, k=3. The direction of time flows inward towards the attractor, then clockwise. One complete set of equivalent trees is shown in each case, and just the nodes of Garden-of-Eden (leaf) states. Data for each basin is provided as follows: attractor period=p, volume=v, leaf density=d, longest transient=t, max in-degree=P_m.
topleft: rule 250, Z_L=0.5, Z_R=0.5, too convergent for encryption, p=1, v=32767, d=0.859, t=14, P_m=1364 (seed=110100101011000).
right: rule 110, Z_L=0.75, Z_R=0.625, too convergent for encryption, p=295, v=10885, d=0.55, t=39, P_m=30 (seed=110100101011000).
bottom: rule 30, a chain-rule, Z_L=0.5, Z_R=1, OK for encryption, p=1455, v=30375, d=0.042, t=321, P_m=2 (seed=110110111000000).

and one's own thumb print, and given that each unique string is linked somewhere within the graph according to a dynamical rule, this immediately suggested that a string with some relevant information could be recovered from another string, linked to it in some remote location in the graph, for example by running backwards from string A (the information) to arrive after a number of time-steps at string B (the encryption), then running forwards from B back to A to decrypt (or the method could be reversed) – so here was a new approach to encryption where the rule is the encryption key, previously described in [7, 8, 11].

Gutowitz patented analogous methods using dynamical systems, CA in particular [1], but these are different from the methods I will describe, where its crucial to distinguish a type of CA rule were the graph linking state-space has the appropriate topology to allow efficient encryption/decryption.

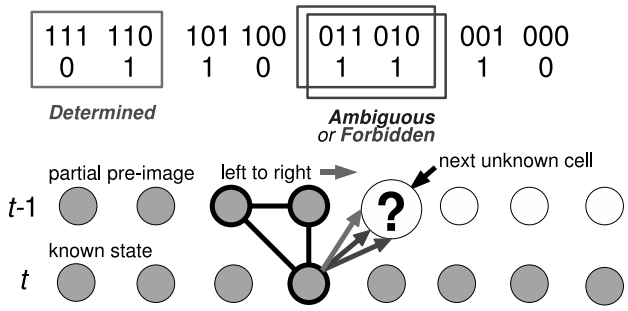

Fig. 4. The CA reverse algorithm, where there are three possibilities for next unknown cell (from left to right in this example): determined, ambiguous or forbidden. The rule illustrated is the 3 neighbour (k=3) rule 110, the decimal equivalent of the rule-table 01101110, with the neighbourhoods ordered according to Wolfram's convention[3], the highest value neighbourhood on the left. Note that the rule-table resolves into adjoining pairs, where the neighbourhoods differs only by their rightmost bit. If the bits in a pair differ, the next cell is determined – if the same, the next cell is either ambiguous or forbidden.

2. The CA reverse algorithm and Z-parameter

Given an arbitrary 1D CA state, a reverse algorithm introduced in [4] and implemented in DDLab [10], can directly compute all of a state's immediate pre-images, or determine that there are none. The algorithm is illustrated in Fig. 4. It works from either left to right or right to left, to find the next unknown cell in a partially known pre-image. There are three possibilities derived from the automata rule - the next cell is: *determined* (so fill in the value and continue to the next), *ambiguous* - 0 and 1 are both valid (so continue from both solutions - the pre-image bifurcates), or *forbidden* (there is no valid solution so halt and abandon this partial pre-image). Once a pre-image is computed periodic boundaries need to be checked – does the value at cell 0 equals the value at cell n+1? otherwise the pre-image is invalid.

Ordering the rule-table according to Wolfram's convention [3], and computing from left to right, adjoining pairs of outputs, where the neighbourhoods differ only by their rightmost bit,[a] conveniently provide the solution for the next cell. If the output bits in a pair differ, the next cell is determined, if the same, the next cell is either ambiguous or forbidden. So the frequency of unequal pairs should provide some insight into the probability of finding determined next cells, which limit bifurcation thus the proliferation of pre-images. A converse procedure applies from right to left.

This is the basis of the Z-parameter, introduced in [4] and generalised in [5, 6]. The fraction of "deterministic pairs" (Z_L for left to right, and Z_R for right to left) in the lookup-table (the CA rule) gives a first approximation of the probability that the next bit is uniquely determined. The complete calculation of Z takes into

[a]There is a converse procedure from right to left, taking pairs where the neighbourhoods differ only by their leftmost bit. To maintain adjoining pairs each neighbourhood could be reflected and the outputs rearranged as in Fig. 5 (but not done in Fig. 8).

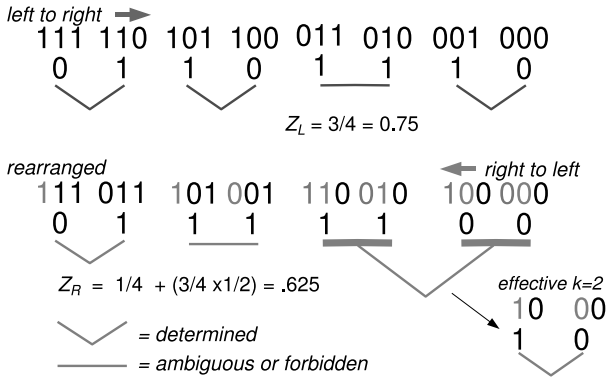

Fig. 5. Calculating the Z-parameter for rule 110. For Z_L there are 3/4 deterministic pairs, indicated with ﹀ links, so Z_L=0.75. For Z_R, where the rule-table has been rearranged by reflection, there are 1/4 deterministic pairs - but the 2 pairs indicted by the ﹀ link must be accounted for because their neighbourhood's leftmost bit is irrelevant - they are effectively k=2 neighbourhoods, so $Z_R = 1/4 + (1 - 1/4) * 2/4 = 0.65$. The Z-parameter = 0.75, the greater of Z_L and Z_R.

account neighbourhoods where effective-k<k (Fig. 5). Being a probability the value of Z_L and Z_R ranges between 0 and 1, and the greater of the two was deemed to be the final Z-parameter.

Z did a good job of predicting the bushiness or branchiness of subtrees in basins of attraction – their typical in-degree (or degree of pre-imaging), which related to the density of end states without pre-images (Garden-of-Eden states) but lets call them "leaves" for short. Low Z, a low probability that the next bit was determined, meant the next bit would probably be both 0 and 1, equally valid, so more pre-images and branchiness, or that there was no valid next bit, more leaves. A branchier tree should have more leaves and shorter branches (transients) because the dynamics leaves behind many states (all those pre-images and leaves) at each time-step, quickly using up the finite state-space. Conversely, high Z, a high probability that the next bit was determined, meant it would probably be either 0 or 1, not both, so fewer pre-images, less branchiness, but more chance of a valid pre-image, so fewer leaves and longer branches. In this case the dynamics leaves behind just a few states at each time-step, using up very little of the finite state-space. Figure 3 gives three examples with contrasting topology.

This nicely tied in with the behaviour of CA when run forward. Low Z, high branchiness results in ordered dynamics. High Z, low branchiness, results in disordered dynamics (chaos) – behaviour that could be recognised subjectively, but also by various objective measures, in particular "input entropy". The entropy stabilises for both order (at a low level) and chaos (at a high level). Entropy that does not stabilise but exhibits variance over time is a sign of complexity [5, 6].

The Z-parameter is related to Langton's λ-parameter [2] which can tune dynamics between order and chaos, with complex dynamics at a phase transition. For binary rules λ simply counts the fraction of 1s in the rule-table. It is an approxima-

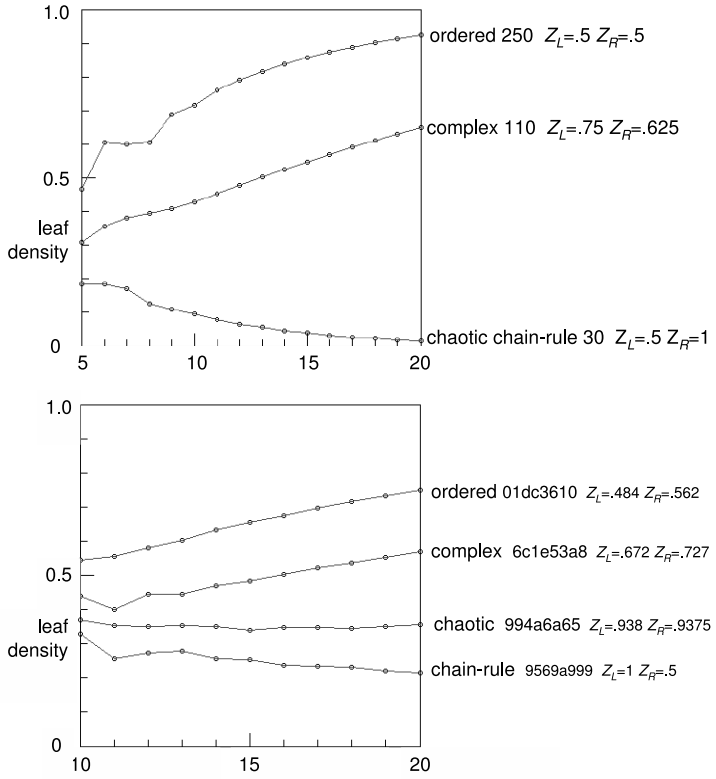

Fig. 6. Plots of leaf density with increasing system size. The measures are for the basin of attraction field, so for the entire state-space. For rule-space in general, leaf density increases with greater n, but for chain-rules leaf density decreases. *above:* k=3, n= 5 to 20, for the three rules in Fig. 3. *below:* k=5, n= 10 to 20, for four typical rules for a range of Z: ordered, complex, chaotic, and maximally chaotic (chain-rule).

tion of Z because as λ approaches 0.5 high Z becomes more likely. Conversely as λ moves away from 0.5 towards 0 or 1 Z is forced lower. However, because Z takes into account the functional distribution of rule-table values, it arguably tracks behaviour more closely [4, 5]. In Fig. 7 the Z-parameter, convergence based on typical in-degree, and the chain-rules, are added to Langton's famous diagram of rule-space based on λ.

3. Limited pre-image rules

Rules in general, even with high values of Z<1, can generate huge numbers of pre-images from typical states, making them inappropriate for encryption, as well as slowing down the reverse algorithm. A branchy (convergent) graph topology has many leaves. As strings become larger, the leaf density increases and usually takes up almost all of state-space (Fig. 6). These leaf states cannot be encrypted by running backward – they do not have pre-images. The alternative, running forwards

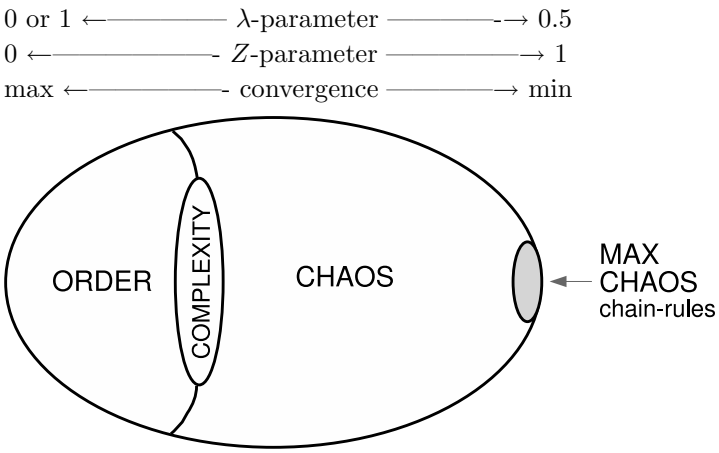

Fig. 7. A view of rule-space (after Langton[2]). Tuning the Z-parameter from 0 to 1 moves the dynamics from maximum to minimum convergence, from order to chaos, traversing a phase transition, where complexity might be found. The chain-rules, added on the right, are maximally chaotic and have the very least convergence, decreasing with system size, making them suitable for dynamical encryption.

to encrypt, then backwards to decrypt, poses the problem of selecting the correct path out of many pre-image branches at each backward time-step. Running forward continually loses information on where the system has come from – CA are dissipative dynamical systems.

But there is a solution! When Z=1, however large the size of the lattice, the number of pre-images of any state becomes strictly limited. Rules where Z=1 can be divided into two types [4] as follows,

- two-way limited pre-image rules: Z_L and Z_R both equal one, where the in-degree must be $\leq 2^{k-1}$.
- one-way limited pre-image rules: Z_L=1 or Z_R=1, but not both, where the in-degree must be less than 2^{k-1} (these are the chain-rules).

This was demonstrated in [4]. Partial pre-images of length k-1 (start-strings) must be assumed in order to generate pre-images, so there are 2^{k-1} possible start-strings. For two-way Z=1 rules there will be at the most just one pre-image per start-string, so the maximum number of pre-images is 2^{k-1}.

Z=1 rules have by definition an equal number of 1s and 0s, so for one-way Z=1 rules (chain-rules) where, say $Z_L < 1$, for every non-determined pair of neighbourhoods with an output of 0, there must be another non-determined pair with an output of 1, because Z_R=1. It follows that whatever the first bit of any state (0 or 1), at least one (left to right) start-string must be invalid, so the number of pre-images must be less than 2^{k-1}. By the same token, if computing from right to left for the same rule where Z_R=1, there must be at least one pre-images where periodic boundaries will not be satisfied.

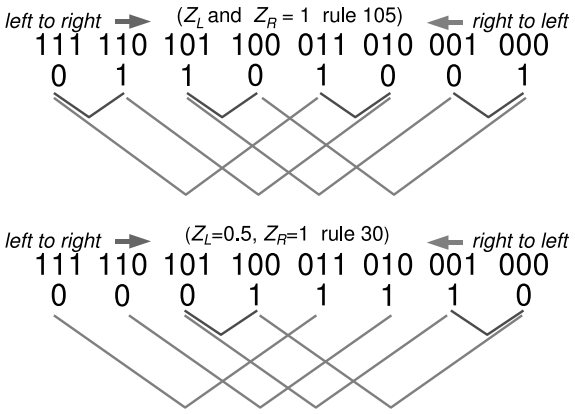

Fig. 8. Two-way and one-way limited pre-image k=3 rules, where $Z = 1$. Determined left to right pairs are indicated by small $\diagup$ links, right to left by large $\diagup$ links. *above:* The two-way limited pre-image rule 105: Z_L and Z_R both equal one. In-degree must be $\leq 2^{k-1}$ – not suitable for encryption. *below:* The one-way limited pre-image chain-rule 30: Z_L=0.5, Z_R=1. The in-degree must be $< 2^{k-1}$. As the system size n increases, the fraction of indegree=1 also increases, so the leaf density decreases making these rules suitable for encryption. To construct a chain-rule, choose the direction, then allocate unequal pairs at random.

For example, when computing pre-images of a known state S from left to right for rule 30 (Fig. 8-*below*), a leftmost cell of S equal to 1 is incompatible with the start string 11$\star$ ($\star$ is a wild-card), a leftmost cell of S equal to 0 is incompatible with the start string 01$\star$. Computing from right to left, the same argument applies to the last (also leftmost) cell of S, 1 and 0, being incompatible with the same left to right start strings 11$\star$ and 01$\star$, so incompatible with periodic boundaries.

Limited pre-imaging (in-degree) appears to produce a promising topology on which to implement encryption, because we would usually need a long string to encode information, but the "two-way" Z=1 rules still suffer from some of the same problems as rules in general, too branchy and a high proportion of leaves. One-way Z=1 rules on the other hand, seem to provide the ideal graph topology. They have an unexpected and not fully understood property: that although the maximum number of pre-images (P_m) of a state must be less than 2^{k-1}, experiment shows that the actual number is usually much less, and decreases as the system size increases; consequently the leaf-density also decreases as in Fig. 6 rule 30.

For large strings of 1000+, P_m=1, except for very rare states where the in-degree=2 in transients, or where transients joint the attractor cycle, which may be extremely long. However, this branching is not a problem when running forward to decrypt, because forward paths converge, and the original pattern will be found. Because the vast majority of state-space occurs in long chains, I renamed the "one-way limited pre-image rules" – "chain-rules".

In Figs. 9 and 10, where k=7, P_m must be less than 2^6=64, but nearly all basins in the basin of attraction field (for a chain-rule constructed at random) have P_m=5

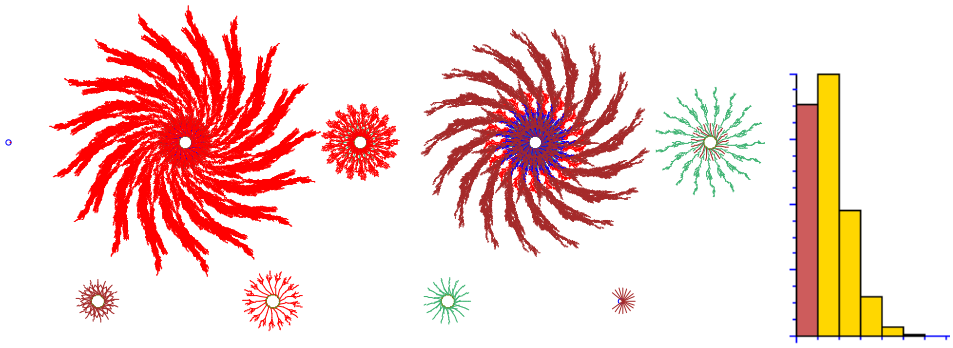

Fig. 9. The basin of attraction field of a chain-rule showing all 9 basins of attraction (state transition graphs) for the k=7 rule (hex)879ac92e2b44774b786536d1d4bb88b41d. Only the links are show, not the nodes. Note there is a tiny attractor (top left) consisting of just one state, all-0s. The chain-rule (Z_L=0.59, Z_R=1) was constructed at random. The string length n=17, state-space=2^{17}=131072, leaf density=0.345, P_m for each basin of the 9 basins is [1,5,5,5,3,4,4,3,18]. *right*: A histogram of in-degree distribution (0 to 6 x-axis), against the fraction of state-space having each in-degree, with in-degree=1 the most abundant.

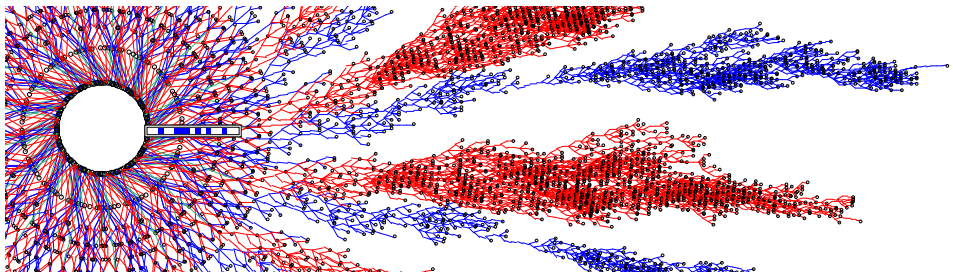

Fig. 10. A detail of the largest basin in Fig. 9 with just the attractor and leaf nodes visible, and one attractor state highlighted. Data is as follows: period=357, basin volume=91868, 70.1% of state-space, leaf density=0.345, max levels=119, P_m=5.

– usually the in-degree is less, as shown in the in-degree distribution histogram Fig. 9 (*right*), though there is a small basin where P_m=18.

As n increases P_m decreases. In Fig. 11 where n=400, P_m=2, but 97% of states have an in-degree of one. As n increases further, in-degrees in subtrees of 2 or more, and leaves, become increasingly rare, and its conjectured that their frequency becomes vanishingly small in the limit.

4. Constructing chain-rules at random

To recap the procedure for finding the Z-parameter from a rule-table in Sec. 2 and Fig. 5, as a first approximation consider pairs of neighbourhoods that differ only by their rightmost bit, so the k-1 bits on the left are the same. Then look at the outputs of these pairs of neighbourhoods in the rule-table to see if they are the same (00 or 11) or different (01 or 10). Z_L is the fraction of different pairs in the

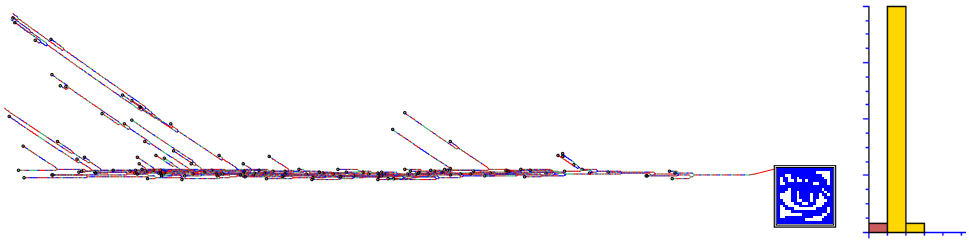

Fig. 11. A subtree of the same chain-rule as in in Figs. 9 and 10, but with a larger system size n=400. The root state (the eye) is shown in 2d (20×20). Backwards iteration was stopped after 500 reverse time-steps. The subtree has 4270 states. Only the 143 leaves are shown as nodes. Leaf density = 0.033, P_m=2 and the density of these branching states is 0.034. *right*: A histogram of in-degree distribution (0 to 5 x-axis), against the fraction of state-space having each in-degree, with in-degree=1 (93%) the most abundant.

look-up table. If all the pairs are different then Z_L=1. Z_R is given by the converse procedure. This is the first approximation, as there are refinements if effective-k in parts of the rule-table is less than k, but we will not bother with those because the pairs procedure gives the most chaotic dynamics.

To assign a chain-rule at random, first pick the direction, Z_L or Z_R at random, then randomly assign different pairs of outputs (01 or 10) to the pairs of neighbourhoods defined above. Check the Z-parameter. If Z_L=1 or Z_R=1 but not both, we have a chain-rule.

From experiment, the lesser value should not be too low, 0.5 or more [7] for binary rules. This is to avoid a gradual lead-in and lead-out of structure in the space-time pattern when decrypting. Ideally the message, picture, or information, should pop out suddenly from a stream of chaos then re-scramble quickly back into chaos, as in Figs. 13 and 15. This is accompanied by a lowering blip in the high input-entropy, the duration of the blip would ideally be minimised to best "hide" the message.

DDLab [10] can assign a chain at random as above, instantaneously, with a key press, see the DDLab manual [7], section 16.7 and elsewhere.

5. How many chain-rules?

How many chain-rules, C, are there in a rule-space $S = 2^{2^k}$? The number of ways (say Z_L=1) "pairs" can be assigned is $2^{2^{k-1}} = \sqrt[2]{S}$. Adding the same for Z_R=1, the number of chain rules C=2($2^{2^{k-1}}$), but we must subtract the number of rules where both Z_L=1 and Z_R=1, which is about $2^{2^{k-2}}$ because pairs in this case are assigned to half of the rule-table and their compliment to the other half. Subtracting also cases where the lesser value of Z is less than 0.5, a round estimate for the number of acceptable chain-rules is $2^{2^{k-1}}$ or the square root of rule-space. This is sufficiently large to provide a virtually inexhaustible supply of keys, for example for k=5: 2^{16}, k=6: 2^{32}, k=7 :2^{64}, k=8: 2^{128} etc. A chain-rule constructed randomly in DDLab will very probably be a unique key.

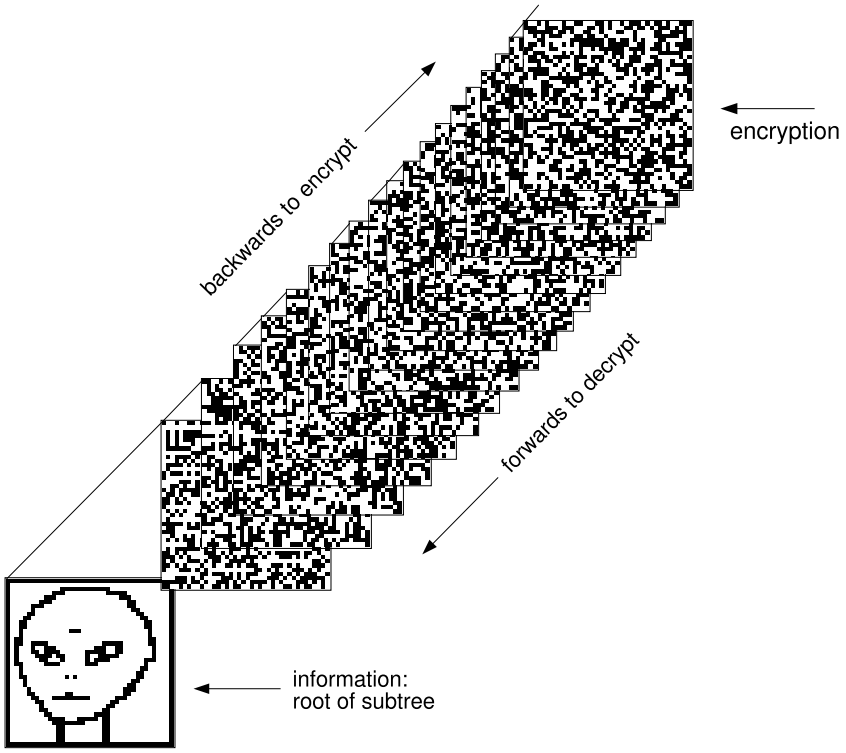

Fig. 12. A subtree applied to encrypt information at its root state (the seed), set to stop after 19 backward time-steps, where the state reached is the encryption. The root state is a 1d bit-pattern, here displayed in 2d (n=1600, 40×40). The "alien" seed was drawn with the drawing function in DDLab. The seed could also be an ASCII file, or any other form of information. With the same k=7 chain-rule as in Fig. 9 the subtree was generated with the CA reverse algorithm. Note the expected lack of branching – highly unlikely because of the large system size. To decrypt, run forwards by the same number of time-steps.

6. Encryption/decryption with chain-rules

The CA reverse algorithm is especially efficient for chain-rules, because the rules-tables are composed purely of determined pairs – they lack the ambiguous pairs that can slow down the reverse algorithm [4]. Many experiments have confirmed that chain-rules make basin of attraction topologies that have the necessary properties for encryption where nearly all states have predecessors and are embedded deeply within long chain-like chaotic transients.

There must still be leaves and states close to the leaves, patterns that could not be encrypted by that particular chain-rule because a backwards trajectory would stop prematurely. However, for big binary systems as in Figs. 12, the state-space is so huge (2^{1600}) that to stumble on an unencryptable state would be very unlikely, but if it were to happen, simply construct a different chain-rule.

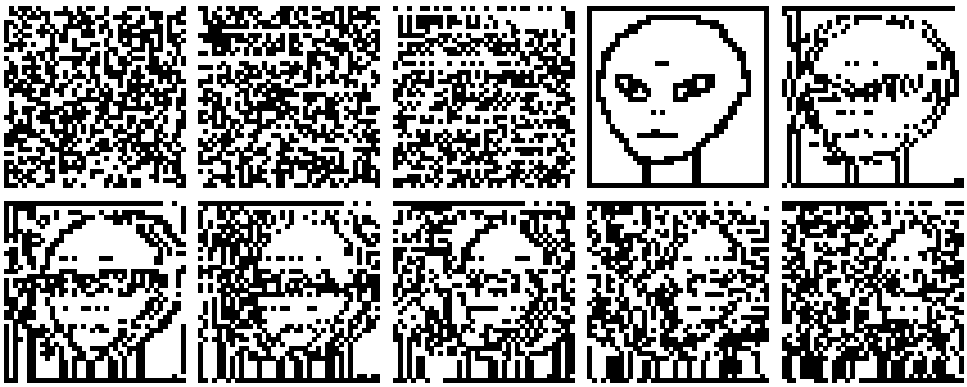

Fig. 13. To decrypt, starting from the encrypted state in Fig. 12 (n=1600, 40×40), the CA with the same chain-rule was run forward by 19 time-steps, the same number that was run backward, to recover the original information. This figure shows time-steps -3 to +6 to illustrate how the "alien" image was scrambled both before and after time step 0.

Encryption/decryption has been available in DDLab since about 1998. To encrypt, select a chain-rule and save it. Select a large enough 1d lattice, which could be shown in 2d. Select the information to be encrypted by loading an ASCII file (for text), or a DDLab image file as the subtreet root (the seed), or create a picture with DDLab's drawing function. Select a subtree, and set it to stop after a number of backwards time-steps. The subtree is unlikely to branch, but branching need not be a significant problem – though it would make more than one possible encryption of the same information. Save the (encrypted) state reached. Fig. 12-15 show examples.

To decrypt, reset DDLab to run forward. Keep the same rule or load it. Load the encrypted state as the initial state. If decrypting a picture, set the presentation for 2d. Run forward, which can be done with successive key-press to see each time-step at leisure. At the 20th time-step, the image will pop out of the chaos. To fully observe the transition, continue stepping forward until the pattern returns to chaos. There will be some degree of ordering/disordering on either side. Figs. 12-15 show examples.

7. Generalising the methods to multi-value

In 2003 all functions and algorithms in DDLab were generalised from binary, v=2 (0,1) to multi-value. The "value-range" v, the number of values or colours, can be set from 2 to 8, i.e. v=3 (0,1,2), v=4 (0,1,2,3), up to v=8 (0,1,2,..7). This included the reverse algorithm, the Z-parameter, chain-rules, and encryption. Details of the new algorithms will be written up at a later date. Any available value-range can be used for encryption, but for efficiency's sake, not to waste bits, v=2, v=4 or v=8 are preferable.

The examples in Figs. 14 and 15 show the encryption of a portrait (v=8, k=4)

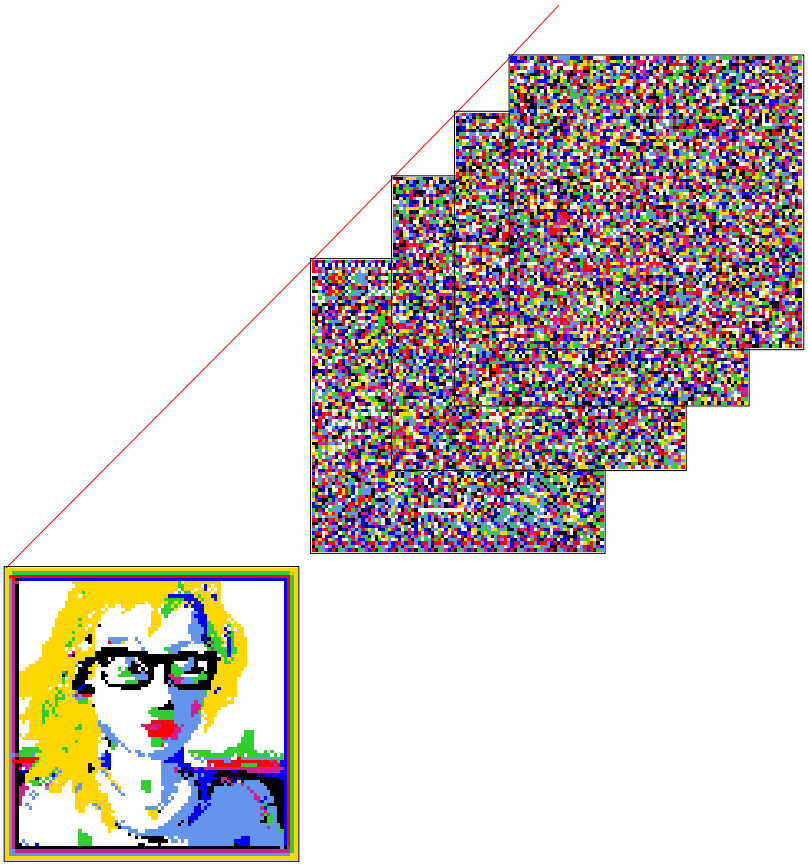

Fig. 14. A 1D pattern is displayed in 2D (n=7744, 88×88). The "portrait" was drawn with the drawing function in DDLab. With a v=8, k=4 chain-rule constructed at random, and the portrait as the root state, a subtree was generated with the CA reverse algorithm as in Fig. 12. The subtree was set to stop after 4 backward time-steps. The state reached is the encryption. To decrypt, run forwards by the same number of time-steps.

on a 88×88 lattice (n=7744), but as v=8 the size of the binary string encoding the lattice is 61052. The v=8 chain-rule was constructed at random; Z_L=0.4, and Z_R=1. When decrypting, the portrait pops out suddenly from chaos, but it takes about 50 time-steps to fully re-scramble back into chaos. This is because k=4 is a small neighbourhood, and the chaotic pattern moves into areas of order at its "speed of light", set by k.

8. Some issues with the method

Chain-rules usually result in attractor cycles with very long periods, though this is relative – the fraction of state-space made up of attractor cycles is probably small. If a state to be encrypted happens to be on an attractor, running it backward may

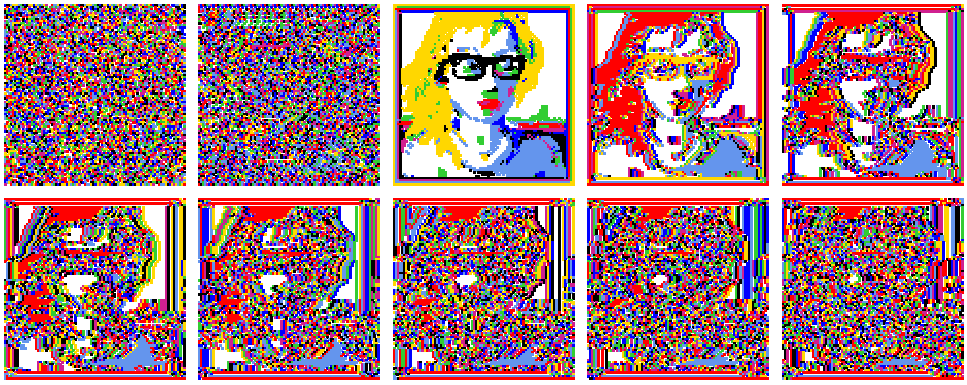

Fig. 15. To decrypt, starting from the encrypted state in Fig. 14 (n=7744, 88×88), the CA with the same v=8 chain-rule was run forward to recover the original image. This figure shows time-steps -2 to +7 to illustrate how the image was scrambled both before and after time step 0.

arrive at a point where a transient joins the attractor. In this case the backwards trajectory will branch.

Note also there is a dynamical law of "rotational symmetry" (and also "bilateral symmetry") that is inescapable in classical CA with periodic boundaries: states with greater rotational symmetry must be downstream of states with lesser rotational symmetry, or with none [4], so symmetry cannot decrease in transients and must stay constant in attractors. Consequently the uniform states made up of just one value (all-0s and all-1s in binary) where rotational symmetry $s = n$ must be downstream of all other states in the dynamics (in basins of attraction) and the states like 010101... ($s = n/2$) must be downstream of states like 00110011... ($s = n/4$), which must be downstream of states with smaller s etc. Uniform states must either form uniform attractors (consisting of just uniform states) including point attractors, or if in a transient must lead directly to another uniform state, then to a uniform attractor. However, these highly ordered states hold little or no information, so are irrelevant for encryption.

The methods described belong to secret key cryptography where the same key (the chain-rule) is used both to encrypt and decrypt, as opposed to public-key cryptography which uses different keys, so he key must be transmitted somehow – and with perfect accuracy. However, the encryption itself could tolerate a degree of noise or errors in transmission because errors will spread during decryption only at the "speed of light" depending on k and the number of forward time-steps.

9. Conclusions

I have described a method of encryption where the key is a chain-rule, a special type of maximally chaotic 1D CA. Information is encrypted by using the key to run the CA backward in time. A secret message can be transmitted openly. The receiver has the same key, and uses it to decipher the message by running the CA forward

in time by an equal number of steps. Anyone could construct their own unique key instantaneously from a virtually unlimited source – its size is about the square root of rule-space.

What is important to someone trying to crack an intercepted encrypted message, with DDLab available? The key itself is vital – data on the CA, its neighbourhood k, and value-range v, is important – not obvious from the key itself. The number of time-steps are useful to know when the forward run should stop. Suppose both the raw information and the encryption where known, could the key be deduced? I do not see a short cut if the two are separated by a number of time-steps, without knowing the intervening steps – massive and exhaustive number crunching would be required.

In other security measures, the key itself could be encrypted. A message could be doubly or multiply encrypted with more than one key. Although these methods are available in DDLab, dedicated software and hardware could be developed for the whole procedure to be fast, hidden and automatic, and also to handle data streams in real time.

References

[Note] For publications by A.Wuensche refer to
www.cogs.susx.ac.uk/users/andywu/publications.html for download.

[1] Gutowitz, H., Method and Apparatus for Encryption, Decryption, and Authentication using Dynamical Systems, U.S. patent application (1992)

[2] Langton, C.G., Computation at the edge of chaos: Phase transitions and emergent computation", Physica D, 42, 12-37, 1990.

[3] Wolfram, S., Statistical mechanics of cellular automata, Reviews of Modern Physics, vol 55, 601–644, 1983.

[4] Wuensche, A., and M.J. Lesser. The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA, 1992.

[5] Wuensche, A., Complexity in 1D cellular automata; Gliders, basins of attraction and the Z parameter, Santa Fe Institute Working Paper 94-04-025, 1994.

[6] Wuensche, A., Classifying cellular automata automatically; finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter, Complexity, Vol.4/no.3, 47–66, 1999.

[7] Wuensche, A., The DDLab Manual (for ddlabx24), section 16.7, Setting a chain rule, www.ddlab.org, 2001.

[8] Wuensche, A., Encryption using Cellular Automata Chain rules, unpublished, 2004.

[9] Wuensche, A.,Discrete Dynamics Lab: Tools for investigating cellular automata and discrete dynamical networks, updated for multi-value, Section 23, Chain rules and encryption, in Artificial Life Models in Software, eds. A.Adamatzky and M.Komosinski, chapter 11, 263-297, Springer. 2005

[10] Wuensche, A., Discrete Dynamics Lab (DDLab), software for investigating discrete dynamical networks, www.ddlab.org (1993-2008)

[11] Wuensche, A., Encryption using cellular automata chain-rules, AUTOMATA-2008: Theory and Applications of Cellular Automata, eds A.Adamatzky et al, Luniver Press, 126–137, 2008.