Encryption using Cellular Automata Chain-rules

Andrew Wuensche, Feb 2004

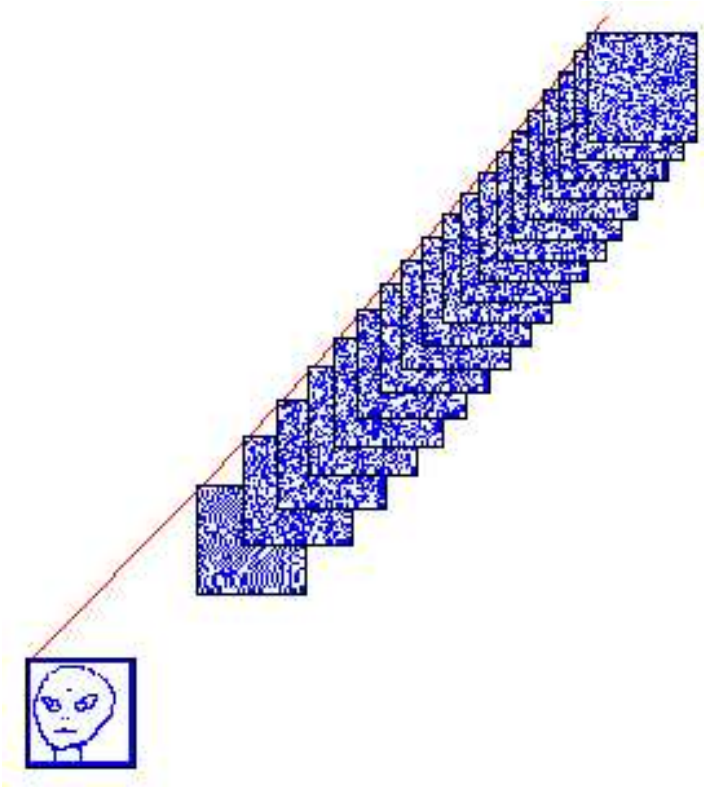

Figure 1: With the "alien" as the root state, a subtree was generated with the reverse algorithm in DDLab; note that the subtree has no branching, and branching is highly unlikely to occur. The rule was a $k = 7$ chain-rule, selected at random (in hex: **a7 4e b6 6b b9 c3 a0 81 58 bi 49 94 46 3c 5f 7e**), with Z-parameter values: $Z_{left} = 0.617$, $Z_{right} = 1$. The subtree was set to stop after 20 backward steps. The 1d CA is displayed in 2d.

The encryption method is implemented in a one-dimensional cellular automata (1d-CA), a well studied discrete dynamical system, were each "cell" in a ring of cells updates its value (0,1) as a function of the values of its k neighbors. All cells update synchronously - in parallel, in discrete time-steps, moving through a deterministic forward trajectory. Each "state" of the ring has one successor, but may have multiple or zero predecessors.

Encryption using 1d CA relies on two methods of my own invention: a "reverse algorithm" for running 1d-CA backwards to compute predecessors, and the Z-parameter which is derived from the reverse algorithm. Both are explained and defined in published work[1, 2], and implemented in DDLab[3]. Combining these ideas to define the a subset of CA rules suitable for encryption is novel.

Consider CA rules with a Z-parameter = 1, where Z_{left}=1 or Z_{right}=1, but not both. Such rules have maximally "chaotic" dynamics[1, 2], with very low convergence, where the maximum in-degree (the number of predecessors) of any state must be less than 2^{k-1} irrespective of the system size, where k is the neighborhood size. I will call such rules chain-rules for reasons that will become clear.

The dynamics for most CA rules is highly convergent, with high in-degree, and an overwhelming fraction of state-space made up of global states without predecessors, the leaves of sub-trees,

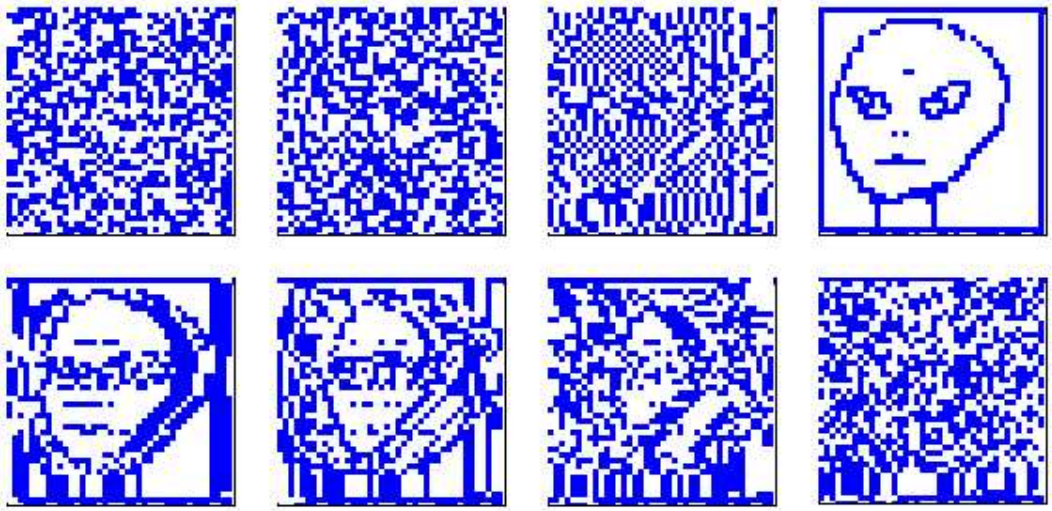

Figure 2: To decrypt, Starting from the encrypted state in figure 1, the CA with the same rule was run forward by 20 time-steps, the same number that was run backwards, to recover the original image or bit-string. This figure shows time-steps 17 to 25 to illustrate how the "alien" image was scrambled both before and after time-step 20.

so called "garden-of-Eden states", whose density increases with system size. Most CA rules are therefore unsuitable for encryption.

By contrast, the typical in-degree of chain-rules is extremely low, usually just one. These rules give basins of attraction with a very low garden-of-Eden density, *decreasing* with system size, and becoming order zero in the limit, so nearly all states have predecessors, and are embedded deeply along transients.

Running the dynamics backwards from a global state, whatever the system size (using the reverse algorithm implemented in DDLab[3], will very efficiently generate a chain-like transient subtree with very rare branching events. A chain-rule will be the encryption key, expressed, like any CA rule, as a lookup-table with 2^k bits.

Rule-space, $S = 2^{2^k}$. Chain-rules form a small subset[1] of S, on the order of $C = 2^{2^{k-1}}$. C is sufficiently large to provide an inexhaustible supply of keys for larger k, for example for k=5: 2^{16}, k=6: 2^{32}, k=7: 2^{64}, k=8: 2^{128}etc. A chain-rule can be constructed randomly in DDLab, providing a probably unique key.

To encrypt information of any kind which can be expressed as a bit-string, for example text in ascii, the bit-string is made the root state of a subtree. Using a chain-rule, the CA is run *backwards* to compute the subtree for a given number of backward time-steps, T. The state (or states) reached is the encrypted message. If the subtree branches there will be more than one valid encryption.

The recipient of the message must have the key, the chain-rule used to encrypt, including the neighborhood size k, and the number of time-steps, T. Using this rule, the CA is run *forwards* by T time-steps. The state reached is the decrypted message.

[1] A better estimate is, $C = 2(2^{2^{k-1}}) - 2^{2^{k-2}}$, where the number of rules with both Z_{left}=1 and Z_{right}=1 is $2^{2^{k-2}}$. These latter have maximum in-degrees of 2^{k-1}, which is too high for encryption.

Conversely the message can be encrypted by running forwards, and decrypted by running backwards. If the subtree branched (a rare event), there would be more than one decrypted message, where all but one would be nonsense.

Note that a chain-rule that is run either forwards or backwards will scramble any message into a chaotic pattern, with very high block entropy, in just a few time-steps. Breaking the encryption would be extremely hard. Having an example of both a message and its encryption would not be much help to uncover The key itself could be encrypted. A message could be doubly or multiply encrypted with more than one key by encrypting the encryption, and correspondingly decrypted.

These methods can currently be demonstrated in DDLab, and can be implimented for multi-value as well as binary cellular automata. Dedicated software based on the existing algorithms could be developed to make the whole procedure hidden and automatic, and also to handle data streams in real time.

References

[1] Wuensche,A., and M.J.Lesser,(1992), "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.

[2] Wuensche,A., (1999) "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. ftp://ftp.cogs.susx.ac.uk/pub/users/andywu/papers/cplex.pdf

[3] Wuensche,A., (1994-2004) "Discrete Dynamics Lab" (DDLab), software for investigating discrete dynamical networks. `www.ddlab.com`

Andrew Wuensche
Discrete Dynamics Inc.
7 Calle Andreita, Nambe, Santa Fe, NM 87506
tel: 505 455 7330, fax: 455 7343
`andyw@cybermesa.com`, `www.ddlab.com`