A Cryptosystem Based on the Composition of Reversible Cellular Automata

Adam Clarridge and Kai Salomaa

Technical Report No. 2008-549
Queen's University, Kingston, Canada
{adam, ksalomaa}@cs.queensu.ca

Abstract. We present conditions which guarantee that a composition of marker cellular automata has the same neighbourhood as each of the individual components. We show that, under certain technical assumptions, a marker cellular automaton has a unique inverse with a given neighbourhood. We use these results to develop a working key generation algorithm for a public-key cryptosystem based on reversible cellular automata originally conceived by Kari. We conclude with a discussion on security and practical considerations for the cryptosystem and give several ideas for future work.

Key words: public-key, cryptography, encryption, reversible, invertible, cellular automata, composition

1 Introduction

Cryptography has been a part of our everyday lives for some time now. Most widely-used public-key encryption algorithms rely on advanced number theoretic results to achieve a high level of security, such as RSA, whose security is believed to rely on the hardness of the integer factorization problem. These systems tend to have relatively slow implementations [12], and since we will always want more efficient and secure encryption algorithms, it makes sense to consider alternate techniques. Cellular automata (CA) as a medium for encryption is an attractive idea in theory because most CA can be implemented on very fast hardware [4, 5, 14], hence a CA-based scheme may have the potential to encrypt and decrypt messages faster than existing techniques.

Most investigations into CA-based cryptosystems have been aimed at traditional secret-key systems [2, 6, 7, 10, 11, 13]. There appear to be very few CA-based public-key cryptosystems in the literature; one is the Finite Automata Public-Key Cryptosystem, or Tao-Chen cryptosystem [12], although it uses non-homogeneous CA. Kari's paper [8] outlines an idea for a public-key cryptosystem based on reversible cellular automata, and poses the question of how to implement the key generation algorithm. We now review this paper in some detail, as it is the main reference for our work.

The general objective of a public-key cryptosystem based on reversible cellular automata is to design an RCA that is very hard to invert without some secret

knowledge. That way, the RCA can be published and its inverse can be kept as the private key. Kari emphasizes the importance that the RCA be at least two-dimensional, since there exist algorithms to invert any one-dimensional RCA [1], and also because of the following theorem.

Theorem 1. [9] *It is undecidable if a given two-dimensional CA is reversible. This is true even when restricted to CA using the von Neumann neighbourhood.*

This theorem provides a sound theoretical basis for the security of Kari's public-key cryptosystem [8]. The basic idea outlined in the paper was to compose together several simple and reversible 'marker' CA (which we define in Section 2) in order to form a more complex cellular automaton

$$C = C_n \circ C_{n-1} \circ \cdots \circ C_1,$$

with inverse

$$C^{-1} = {C_1}^{-1} \circ {C_2}^{-1} \circ \cdots \circ {C_n}^{-1}.$$

Encryption occurs by encoding the message as the initial configuration of the CA, then evolving the composed CA for some k generations to obtain the ciphertext. The inverse automaton does not need to be computed explicitly; one need only apply each component of the composition in succession. The inverse is then applied for k iterations to decrypt the ciphertext. The composition $C_n \circ C_{n-1} \circ \cdots \circ C_1$ is the public key, and each of the inverse automata of the composition $({C_1}^{-1}, {C_2}^{-1}, \ldots, {C_n}^{-1})$ are kept as the private key. A well-constructed public key should be very hard to invert without knowledge of the components $C_1, C_2, \ldots, C_n$ because the neighbourhood size of the inverse automaton would be quite large.

The paper [8] includes an example of a marker RCA composition with a 2-dimensional neighbourhood of 4 cells, and whose inverse has a 2-dimensional neighbourhood of 9 cells. The composition is made up of 5 very simple reversible marker CA. This is of course just an illustrative example, and Kari points out that longer and more complex (more states and a less restricted form) compositions would be needed in order to ensure security against brute force attacks. However, a public key with s states and neighbourhood size n requires s^n entries in its local rule table, so it is essential to try to keep n small so that the public key can be stored in reasonably sized memory.

The main issue preventing the practical implementation of Kari's cryptosystem is the question of how to choose (or randomly generate) reversible marker CA such that the neighbourhood size of the composition remains small. In this paper, we give one possible answer to this question and investigate the resulting working cryptosystem.

We will state some preliminary assumptions and definitions before discussing our results concerning the composition of a class of marker CA in Sections 2 and 3. We give an algorithm[1] for generating public and private keys in Section 4, and discuss practical implementation issues, security considerations, and ideas for future research in Section 5.

[1] Email the first author for a working software prototype.

2 Preliminaries

In this paper we assume that in a cellular array containing $M_1 M_2 \cdots M_d$ cells, where M_i is the number of cells of each dimension for $i = 1, \ldots, k$, the neighbours of cells near the edge of the cellular array are determined by adding the component indices cyclically (modulo M_i). This is simply the toroidal boundary condition.

A 'marker' CA is defined by a permutation ϕ of the state set, and a finite collection of patterns $P_1, P_2, \ldots, P_k$ around the origin, where each P_i is a mapping from a finite subset X_i of $\mathbb{Z}^d$ into the state set. For each cell c, the local rule of the marker CA checks if any of the patterns $P_1, P_2, \ldots, P_k$ is present as the neighbourhood of c. If so, the permutation ϕ is applied to c, and if not, then c's state does not change. Marker CA are also known as 'marker automorphisms of the one-sided d-shift' [3] in the dynamical systems literature.

We define a 'fixed-domain' marker cellular automaton (or FDM CA) to be a five-tuple (d,S,N,A,f) with dimension d, state set S, neighbourhood vector

$$N = (\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k),\ \bar{n}_i \in \mathbb{Z}^d \qquad \text{for } i = 1, 2, \ldots, k,$$

acting set $A \subseteq S^k$ with entries corresponding to the positions defined by N, and a function $f : S \to S$. The local rule of an FDM CA acts on a cell c (in state s) in the following simple way: if the neighbours of c are in a state configuration corresponding to an element of A, then the state of c on the next generation is $f(s)$. Otherwise, the state of c does not change. An FDM CA is just a special type of marker CA where all of the patterns are mappings from N to S, hence the term 'fixed-domain'. Note that, conversely, an arbitrary marker CA can be represented as an FDM CA by choosing N to be sufficiently large.

In the next section we give necessary and sufficient conditions characterizing change in neighbourhood size of compositions of FDM CA. For this purpose we do not need to assume that f is one-to-one, however, when the FDM CAs are required to be reversible, it is necessary (though not sufficient to guarantee invertibility) for f to be one-to-one.

In this paper, we use the terms 'invertible' and 'reversible' interchangeably when referring to cellular automata. Also we define compositions of cellular automata in the following way: for any two cellular automata C_1 and C_2 acting on the same cellular grid, one generation of the CA $C_2 \circ C_1$ refers to the application of one generation of C_1 followed by one generation of C_2.

3 Theoretical Results

3.1 Neighbourhood Size of Compositions

As we have noted above, for implementing a public-key cryptosystem based on compositions of RCAs, a desireable property is that the composition should have a small neighbourhood size. Here we give necessary and sufficient conditions that characterize the effect on neighbourhood size of composing an FDM CA with an

arbitrary CA. For readability, we give the result first for a very restricted type of CA with a single cell neighbourhood. The underlying idea used for the general case (in Proposition 2) is similar but the notation is more complicated.

Let B be a CA with dimension $d = 1$, state set S, neighbourhood $N = (-1)$ (the cell to the left), and arbitrary transition function.

The state changes of B can be described by a function $h_B : S \times S \to S$. If s is the state of a cell c at time t, then at time $t+1$ the state of c will be $h_B(s', s)$, where s' is the state of the left neighbour of c at time t.

For $s \in S$ we denote

$$\text{next_state}_B(s) = \{h_B(s', s) \mid s' \in S\}.$$

The set $\text{next_state}_B(s)$ consists of all possible states that a state s may directly transition into (depending on the left neighbour of s).

We want conditions which guarantee that the composition of B with an FDM CA that has the same dimension, state set, and neighbourhood as B has the same neighbourhood as B.

Proposition 1. *Let B be an arbitrary CA with dimension $d = 1$, state set S, a neighbourhood $N = (-1)$ (the cell to the left) and transition function h_B. Let D be an FDM CA (d,S,N,A_D,f_D).*

The composition $D \circ B$ has neighbourhood N if and only if for all $s \in S$,

$$(\exists s' \in S) : f_D(h_B(s, s')) \neq h_B(s, s') \Rightarrow \begin{array}{l} \text{next_state}_B(s) \subseteq A_D \text{ or} \\ \text{next_state}_B(s) \cap A_D = \emptyset \end{array} \quad (1)$$

Proof. Suppose that condition (1) holds. Consider two consecutive cells c_1 and c_2 that are in states s and s' respectively. We have to show that when applying the composition $D \circ B$, the next state of c_2 depends only on s and s'. If $f_D(h_B(s, s')) = h_B(s, s')$, $D \circ B$ maps the state of c_2 always to $h_B(s, s')$. Next assume that $f_D(h_B(s, s')) \neq h_B(s, s')$. Now according to (1), $\text{next_state}_B(s)$ is either a subset of A_D or it is disjoint with A_D. In the former case, independently of the state of the left neighbour of c_1, $D \circ B$ maps the state of c_2 to $f_D(h_B(s, s'))$. In the latter case, again independently of the left neighbour of c_1, $D \circ B$ maps the state of c_2 to $h_B(s, s')$. Thus, we can compute the transition of $D \circ B$ at cell c_2 knowing just the current states of c_1 and c_2.

Conversely, assume that (1) does not hold. This means that there exist $s, s_1, s_2 \in S$ such that

$$h_B(s_1, s) \in A_D \quad \text{and } h_B(s_2, s) \notin A_D,$$

and that there exists an $s' \in S$ such that $f_D(h_B(s, s')) \neq h_B(s, s')$.

Consider three consecutive cells c_1, c_2, c_3 that at time t are in states s_1, s, s'. Now in $D \circ B$, the B automaton changes the states of c_2 and c_3 to $h_B(s_1, s)$ and $h_B(s, s')$ respectively. Now since $h_B(s_1, s) \in A_D$, the D automaton changes the state of c_3 to $f_D(h_B(s, s'))$. So given the configuration s_1, s, s' at time t, $D \circ B$ maps the state s' to $f_D(h_B(s, s'))$ at time $t + 1$.

On the other hand, if the states of c_1, c_2, c_3 at time t are s_2, s, s', in $D \circ B$, the B automaton changes the states of c_2 and c_3 to $h_B(s_2, s)$ and $h_B(s, s')$ respectively. Since $h_B(s_2, s) \notin A_D$, we know that the D automaton will not change the new state of c_3. So given the configuration s_2, s, s' at time t, $D \circ B$ maps the state s' to $h_B(s, s')$ at time $t + 1$.

Since $h_B(s, s') \neq f_D(h_B(s, s'))$, this means that the CA $D \circ B$ does not have neighbourhood N, since there is a dependency on the neighbour two cells to the left. □

We now address the more general case, where B is an arbitrary cellular automaton with state set S, neighbourhood $N_B = (\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k), \bar{n}_i \in \mathbb{Z}^d,\ d \geq 1$, and local transition function $h_B : S^k \to S$ (h_B maps the neighbourhood of a cell to its next state).

Denote the set of all possible states of the neighbourhood $N_B = (\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k)$ of a cell c by

$$\mathbf{S}_{N_B}(c) = \{(s_{\bar{n}_1}, s_{\bar{n}_2}, \ldots, s_{\bar{n}_k}) \mid s_{\bar{n}_i} \in S \text{ for } i = 1, \ldots, k\},$$

where each $s_{\bar{n}_i}$ refers to the state of the cell in position $\bar{n}_i$. A k-tuple of $\mathbf{S}_{N_B}(c)$ determines, according to the local transition function h_B, the state of the cell c at the next time step.

The neighbourhood of the neighbourhood of a cell c contains any cell that is a neighbour to one of c's neighbours. Let us refer to this set as the *second order* neighbourhood of c. We will assume without loss of generality that each cell is a neighbour to itself, so each cell in the neighbourhood of c belongs to its second order neighbourhood as well.

Denote the collection of all second order neighbourhoods of a cell c with neighbourhood $\mathbf{s} = (s_{\bar{n}_1}, s_{\bar{n}_2}, \ldots, s_{\bar{n}_k}) \in \mathbf{S}_{N_B}(c)$ by

$$\bar{\mathbf{S}}_{N_B}(\mathbf{s}) = \left\{ \begin{bmatrix} t_{\bar{n}_1+\bar{n}_1} & t_{\bar{n}_1+\bar{n}_2} & \ldots & t_{\bar{n}_1+\bar{n}_k} \\ t_{\bar{n}_2+\bar{n}_1} & t_{\bar{n}_2+\bar{n}_2} & \ldots & t_{\bar{n}_2+\bar{n}_k} \\ & \vdots & & \\ t_{\bar{n}_k+\bar{n}_1} & t_{\bar{n}_k+\bar{n}_2} & \ldots & t_{\bar{n}_k+\bar{n}_k} \end{bmatrix} \in S^{k \times k} \;\middle|\; \forall \bar{n} \in N_B, t_{\bar{n}} = s_{\bar{n}} \right\}.$$

The rows of each matrix in $\bar{\mathbf{S}}_{N_B}(\mathbf{s})$ are the neighbourhoods of each of the cells in $\mathbf{s}$. The states in positions $\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k$ are fixed (they are the states of $\mathbf{s}$), while the rest of the second order neighbourhood is arbitrary.

The automaton B's neighbourhood state changes for a cell c with neighbourhood $\mathbf{s}$ can be described by a function

$$\bar{\mathbf{h}}_B : \bar{\mathbf{S}}_{N_B}(\mathbf{s}) \to \mathbf{S}_{N_B}(c)$$

which maps a second order neighbourhood to the next neighbourhood of c. The second order neighbourhood contains all the information needed in order to determine the next neighbourhood of c.

For the neighbourhood $\mathbf{s} \in \mathbf{S}_{N_B}(c)$ we denote the set of all possible next neighbourhoods by

$$\text{next_neighbourhood}_B(\mathbf{s}) = \{\bar{\mathbf{h}}_B(\bar{\mathbf{r}}) \mid \bar{\mathbf{r}} \in \bar{\mathbf{S}}_{N_B}(\mathbf{s})\}.$$

Now we can extend the result of Proposition 1.

Proposition 2. *Let B be an arbitrary CA with dimension d, state set S, neighbourhood N_B, and transition function h_B. Let D be an FDM CA (d,S,N_B,A_D,f_D).*

The composition $D \circ B$ has neighbourhood equal to N_B if and only if for all $\mathbf{s} \in \mathbf{S}_{N_B}(c)$,

$$f_D(h_B(\mathbf{s})) \neq h_B(\mathbf{s}) \Rightarrow \begin{array}{l} \text{next_neighbourhood}_B(\mathbf{s}) \subseteq A_D \text{ or} \\ \text{next_neighbourhood}_B(\mathbf{s}) \cap A_D = \emptyset \end{array} \tag{2}$$

Proof. Suppose that condition (2) holds. We want to show that for all $\mathbf{s} \in \mathbf{S}_{N_B}(c)$, where c is some cell, we do not need any more information than the neighbourhood $\mathbf{s}$ to compute the transition of $D \circ B$. For neighbourhoods $\mathbf{s} \in \mathbf{S}_{N_B}(c)$ such that the left hand side of condition (2) is false, D always maps $h_B(\mathbf{s})$ to itself. Clearly in this case D does not create a dependence on a larger neighbourhood. Now consider neighbours $\mathbf{s} \in \mathbf{S}_{N_B}(c)$ such that both sides of the implication (2) are true. The right side of (2) means that all next possible neighbourhoods of $\mathbf{s}$ must either be contained in the acting set of D, or completely separate from the acting set of D. So the particular neighbourhood that $\mathbf{s}$ actually gets mapped to by B is not important, since D already knows from $\mathbf{s}$ whether or not it will act. If $\text{next_neighbourhood}_B(\mathbf{s}) \subseteq A_D$, then D will apply f_D to $h_B(\mathbf{s})$, and if $\text{next_neighbourhood}_B(\mathbf{s})$ is disjoint from A_D, then D will apply the identity map to $h_B(\mathbf{s})$. Thus, we can compute the transition of $D \circ B$ at a cell c knowing just the current states of its neighbours, $\mathbf{s}$.

Conversely, assume that (2) does not hold. This means that for some $\mathbf{s} \in \mathbf{S}_{N_B}(c)$ there exist $\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2 \in \bar{\mathbf{S}}_{N_B}(\mathbf{s})$ such that

$$\bar{\mathbf{h}}_B(\bar{\mathbf{r}}_1) \in A_D \quad \text{and } \bar{\mathbf{h}}_B(\bar{\mathbf{r}}_2) \notin A_D,$$

and that $f_D(h_B(\mathbf{s})) \neq h_B(\mathbf{s})$. Recall that $\bar{\mathbf{r}}_1$ and $\bar{\mathbf{r}}_2$ agree on the states of $\mathbf{s}$, and that $\bar{\mathbf{h}}_B$ denotes that function that maps a second order neighbourhood to a neighbourhood of B.

Consider a collection of cells in the configuration of $\bar{\mathbf{r}}_1$. When we apply $D \circ B$, the B automaton changes the states of $\mathbf{s}$ to a neighbourhood which is in A_D. The D automaton is then applied. So the next state of the cell c is $f_D(h_B(\mathbf{s}))$.

On the other hand, consider a collection of cells in the configuration of $\bar{\mathbf{r}}_2$. When we apply $D \circ B$, the B automaton changes the states of $\mathbf{s}$ to a neighbourhood which is not in A_D. The D automaton applies the identity map. So the next state of the cell c is $h_B(\mathbf{s})$.

Since $f_D(h_B(\mathbf{s})) \neq h_B(\mathbf{s})$, this means that the CA $D \circ B$ cannot have the neighbourhood N_B, since it depends on one or more of the states of $\bar{\mathbf{r}}_1$ and $\bar{\mathbf{r}}_2$ which differ and are outside of N_B. □

The condition of Proposition 2 can be used to inductively define a sequence of FDM CAs $C_1, C_2, \ldots, C_n$ such that $C_1 \circ C_2 \circ \cdots \circ C_n$ has the same neighbourhood as each of its components.

One interesting property of FDM CAs is that a carefully chosen composition can represent any cellular automaton.

Proposition 3. *Every cellular automaton C with neighbourhood N_C of size k and state space S can be represented as a composition of $|S|^k+1$ FDM CAs with the same neighbourhood, if the FDM CAs are allowed $|S|+|S|^k$ states.*

Proof. We give a proof by direct construction. Consider a cellular automaton C with neighbourhood N_C of size k. The automaton has a state transition function for updating the state of a cell c,

$$f_C : S^k \to S$$

with $|S|^k$ inputs. Enumerate all of these inputs by $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n$, where $n = |S|^k$, and let $\mathbf{m}_i[0]$ refer to the state of c in input $\mathbf{m}_i$. Define $|S|^k$ new states by $p_1, p_2, \ldots, p_n$. The FDM CAs with neighbourhood N_C of size k that will form the composition will use state space $S \cup \{p_1, p_2, \ldots, p_n\}$, where the p states will be used to temporarily refer to states in S.

We now describe the composition of the $n+1$ FDM CAs $D_{n+1} \circ D_n \circ \cdots \circ D_1$ that will emulate C. The first automaton in the composition, D_1, has acting set equal to the singleton $\mathbf{m}_1$, and maps $\mathbf{m}_1[0]$ to the state p_1. Let $\mathbf{m}_2 = (m_2^1, m_2^2, \ldots, m_2^k)$. Then the next automaton in the composition, D_2, has acting set

$$A_{D_2} = \left\{ (s_1, s_2, \ldots, s_k) \in (S \cup \{p_1\})^k \middle| \forall i : \begin{array}{c} s_i = p_1 \Rightarrow m_2^i = \mathbf{m}_1[0] \text{ and} \\ s_i \in S \Rightarrow s_i = m_2^i \end{array} \right\}$$

and maps all states of S to p_2. Note that the composition $D_2 \circ D_1$ has the same neighbourhood N_C of size k; if the neighbourhood of a cell of C is in the configuration of $\mathbf{m}_2$, then by our choice of D_2's acting set, we know we are guaranteed that D_1 will map this configuration to an element of A_{D_2}. Also, if the neighbourhood of a cell is not in the configuration of $\mathbf{m}_2$, then we know that D_1 will not map the neighbourhood anywhere in D_2's acting set. So the condition of Proposition 2 holds, and we have that $D_2 \circ D_1$ has neighbourhood N_C.

The next cellular automata in the composition up to D_n have similar form. Let $\mathbf{m}_j = (m_j^1, m_j^2, \ldots, m_j^k)$, then for all j up to n, D_j has acting set

$$A_{D_j} = \left\{ (s_1, \ldots, s_k) \in (S \cup \{p_1, \ldots, p_{j-1}\})^k \middle| \forall i : \begin{array}{c} s_i = p_1 \Rightarrow m_j^i = \mathbf{m}_1[0] \text{ and} \\ s_i = p_2 \Rightarrow m_j^i = \mathbf{m}_2[0] \text{ and} \\ \vdots \\ s_i = p_{j-1} \Rightarrow m_j^i = \mathbf{m}_{j-1}[0] \text{ and} \\ s_i \in S \Rightarrow s_i = m_j^i \end{array} \right\}$$

and maps always to p_j. In this way, D_j performs the j^{th} state transition, regardless of other cells in the neighbourhood which are changed by other state transitions, since they are changed to members of the temporary set of p states.

The composition $D_n \circ D_{n-1} \circ \cdots \circ D_1$ has neighbourhood N_C, since each automaton which is added to the composition has an acting set which guarantees that the condition of Proposition 2 holds. The last automaton in the composition, D_{n+1}, has acting set equal to $(S \cup \{p_1, p_2, \ldots, p_n\})^k$ (it always acts), and applies the following mapping to the state s of a cell:

$$f_{D_{n+1}}(s) = \begin{cases} f_C(\mathbf{m}_i) \text{ if for some } i,\ s = p_i \\ s \qquad\qquad \text{otherwise} \end{cases}$$

The automaton D_{n+1} maps the temporary p states back to the original state space S. The result is that the composition $D_{n+1} \circ D_n \circ \cdots \circ D_1$ mimics the behaviour of the C automaton exactly for inputs which are from S^*, because of the use of temporary states which store state transition information in a way that does not interfere with other state transitions. □

The upper bounds of Proposition 3 are not meant to be tight at all in terms of the number of cellular automata required in the composition or the number of extra states required. An extension to this work could be to create an algorithm which, for a given cellular automaton, automates this construction and approximates or finds exactly the minimal number of automata and extra states required.

3.2 Reversibility

We now discuss the reversibility of FDM CA. In the following text we use the notation $\mathbf{s}[0]$ to refer to the state in the 'zero' position of a neighbourhood vector $\mathbf{s} \in \mathbf{S}_N$.

Lemma 1. *Let $C = (d, S, N, A_C, f)$ be an FDM CA, and assume the cell itself is part of N $(0 \in N)$ without loss of generality. Denote*

$$B = \{\mathbf{a} \in A_C \mid f(\mathbf{a}[0]) \neq \mathbf{a}[0]\}.$$

Then (d, S, N, B, f) is equivalent with C.

Proof. The proof is immediate because tuples of A_C not in B do not affect the computation in any way. □

We say that an FDM CA C with active set A_C is *reduced* if for every $a \in A_C$, $f(a[0]) \neq a[0]$. By Lemma 1, without loss of generality we can assume that an arbitrary FDM CA is reduced.

We now generalize some of the definitions from Proposition 2 for an arbitrary FDM CA C=(d,S,N,A_C,f).

Let an arbitrary neighbourhood be denoted by N'. Then let the (N, N')-neighbourhood of a neighbourhood $\mathbf{s} \in \mathbf{S}_N$ be the configuration containing the N neighbours of each of the elements in the N' neighbourhood. Let the set of all (N, N')-neighbourhoods of $\mathbf{s}$ be denoted by $\bar{\mathbf{S}}_{(N,N')}(\mathbf{s})$. Note that an N-neighbourhood of an N'-neighbourhood is the same as an N'-neighbourhood of

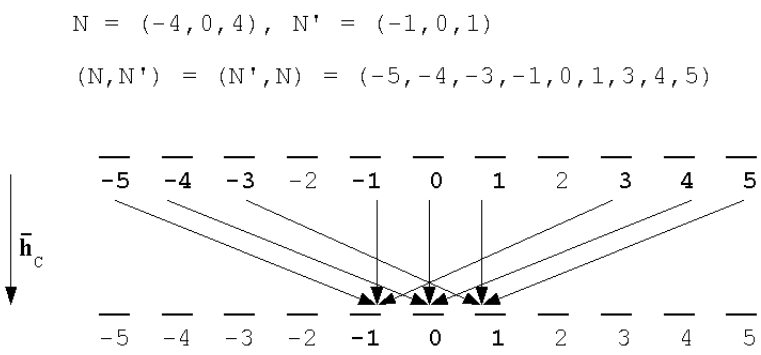

Fig. 1. An example of an (N, N')-neighbourhood mapping to an N' neighbourhood.

an N-neighbourhood, which can be more formally stated as follows. Let r be a vector in $\bar{\mathbf{S}}_{(N,N')}(\mathbf{s})$ and let $r_{N'}$ be the 'restriction' of r to the neighbourhood N', that is, $r_{N'}$ is an N'-neighbourhood around the zero-position. Then the (N', N)-neighbourhood of $r_{N'}$ equals r.

Let the transition function of C from (N, N')-neighbourhoods to N' neighbourhoods be denoted by $\bar{\mathbf{h}}_C$, which takes an (N, N')-neighbourhood configuration and the neighbourhood N' as input, and outputs C's action with that configuration on the neighbourhood of size N'. An illustration of an (N, N')-neighbourhood and how it maps to an N' neighbourhood is given in Figure 1.

Let the transition function from neighbourhoods to sets of possible output neighbourhoods be denoted by

$$\text{next_neighbourhood}_C(\mathbf{s}, N') = \{\ \bar{\mathbf{h}}_C(\bar{\mathbf{r}}, N') \mid \bar{\mathbf{r}} \in \bar{\mathbf{S}}_{(N,N')}(\mathbf{s})\ \}.$$

The following result characterizes when a given FDM CA with neighbourhood N has an FDM CA inverse with neighbourhood N'.

Proposition 4. *Let C be a reduced FDM CA (d,S,N,A_C,f). Denote*

$$X = \bigcup_{\mathbf{a} \in A_C} \text{next_neighbourhood}_C(\mathbf{a}, N'). \tag{3}$$

Then C has an FDM CA inverse with state set S and neighbourhood N' if and only if

$$(\forall\ \mathbf{a} \notin A_C)\ \ f(\mathbf{a}[0]) \neq \mathbf{a}[0] \Rightarrow \text{next_neighbourhood}_C(\mathbf{a}, N') \cap X = \emptyset. \tag{4}$$

Proof. Assume condition (4) holds. Let us choose $C^{-1} = (d, S, N', A_{C^{-1}}, f^{-1})$ where $A_{C^{-1}} = X$, and show that it inverts C. Consider an arbitrary $\bar{\mathbf{r}} \in \bar{\mathbf{S}}_{(N,N')}(\mathbf{a})$ of a neighbourhood $\mathbf{a} \in \mathbf{S}_N$, such that $\bar{\mathbf{h}}_C(\bar{\mathbf{r}}, N') = \mathbf{b}$ $(\in \mathbf{S}_{N'})$. If

$\mathbf{a} \in A_C$, then we know that $\mathbf{b} \in A_{C^{-1}}$ so C^{-1} will map $\mathbf{b}[0]$ to $f^{-1}(\mathbf{b}[0]) = \mathbf{a}[0]$. Now consider the case where $\mathbf{a} \notin A_C$. In this case we know that $\mathbf{a}[0] = \mathbf{b}[0]$. If $f(\mathbf{a}[0]) = \mathbf{a}[0](= \mathbf{b}[0])$, then C^{-1} correctly maps $\mathbf{b}[0]$ back to itself since $f^{-1}(\mathbf{b}[0]) = \mathbf{b}[0]$. On the other hand, if $f(\mathbf{a}[0]) \neq \mathbf{a}[0]$, then from (4) we know that $\mathbf{b} \notin A_{C^{-1}}$, and again C^{-1} must map $\mathbf{b}[0]$ to itself.

Conversely, assume that C has an inverse FDM CA D with neighbourhood N' and let A_D be the active set of D. Since D must correctly 'map back' all states where C applied the function f, it is clear that X (as defined in (3)) is a subset of A_D.

It remains to show that (4) holds. For the sake of contradiction assume that $\mathbf{b} = \bar{\mathbf{h}}_C(\bar{\mathbf{r}}, N') \in X$, where $\bar{\mathbf{r}}$ is an (N, N')-neighbourhood of a neighbourhood $\mathbf{a} \notin A_C$, and $f(\mathbf{a}[0]) \neq \mathbf{a}[0]$. Since $\mathbf{a} \notin A_C$, we know that $\mathbf{a}[0] = \mathbf{b}[0]$. Since D is an inverse of C, the function used by D must be f^{-1}. Since $\mathbf{b} \in X \subseteq A_D$, the FDM CA D applies the function f^{-1} to $\mathbf{b}[0]$, but the result cannot be $\mathbf{a}[0]$ since that would imply $f(\mathbf{a}[0]) = \mathbf{b}[0] = \mathbf{a}[0]$, a contradiction. This means that D is not the inverse of C. □

The following corollary addresses the uniqueness of FDM CA inverses.

Corollary 1. *Let C and X be as defined in Proposition 4, and let C have some inverse C^{-1} with neighbourhood N'. Then C^{-1} is the only reduced FDM CA with neighbourhood N' that inverts C.*

Proof. Any inverse of C must have function f^{-1}. Assume for the sake of contradiction that there exists an FDM CA D with neighbourhood N' that inverts C and has acting set $A_D \neq X$. In the proof of Proposition 4 we have observed that X must be a subset of A_D. Thus it is sufficient to show that there cannot be an element $\mathbf{b} \in A_D$ with $\mathbf{b} \notin X$.

Let $\mathbf{b} = \bar{\mathbf{h}}_C(\bar{\mathbf{r}}, N')$, where $\bar{\mathbf{r}} \in \bar{\mathbf{S}}_{(N,N')}(\mathbf{a})$, $\mathbf{a} \notin A_C$ since $\mathbf{b} \notin X$. Then D cannot be the inverse of C because C maps $\mathbf{a}[0]$ to itself, but D maps $\mathbf{b}[0] = \mathbf{a}[0]$ to $f^{-1}(\mathbf{b}[0])$, which cannot be equal to $\mathbf{b}[0]$ since $\mathbf{b} \in A_D$ and D is reduced. □

4 A Public-Key Cryptosystem

We want to use the idea of composing together many simple RCAs to form a complex RCA that is hard to invert, as outlined in the paper by Kari [8]. In order to make this idea work, we need to have some way to randomly generate a sequence of simple CAs such that the neighbourhood size of their composition remains small (or constant), and each CA in the composition is reversible.

We will demand that the neighbourhood size of each cellular automaton in the composition is the same, and that the entire composition has the same neighbourhood as any of the components. The components will all be FDM CAs. Note that a composition of FDM CAs is not necessarily an FDM CA. Since the neighbourhood, state set, and dimension are fixed, we must design an algorithm which generates acting sets and transition functions for each of the n components

$C_1, C_2, \ldots, C_n$. From the theory in the previous section, we can now state some requirements for such an algorithm.

To maintain neighbourhood size during composition, the FDM CA C_j must have an acting set A_j and transition function f_j such that the composition $C_j \circ (C_{j-1} \circ C_{j-2} \circ \cdots \circ C_1)$ has the same neighbourhood, for all $j \in \{2, \ldots, n\}$. Referring to the condition from Proposition 2, we need to guarantee that for each neighbourhood, the next neighbourhood set of $C_{j-1} \circ C_{j-2} \circ \cdots \circ C_1$ is either completely contained in A_j or is disjoint from A_j. Denote by $T \subseteq S$ the "change set", that is, the set of all states that the composition $C_{j-1} \circ C_{j-2} \cdots \circ C_1$ can possibly change. One way we can be sure to retain neighbourhood size during composition is by setting A_j equal to the set of all neighbourhoods which contain a state in T. The condition from Proposition 2 is satisfied since all neighbourhoods containing states in T will certainly be mapped (by $C_{j-1} \circ C_{j-2} \circ \cdots \circ C_1$) to neighbourhoods which also contain states in T (assuming $f_1, f_2, \ldots, f_{j-1}$ are one-to-one mappings), and neighbourhoods which do not contain any states in T will clearly be mapped to neighbourhoods which do not contain any states in T. We use a less restricted version of this principle (which still satisfies the neighbourhood size preservation condition) in our algorithm to determine the acting set of each FDM CA in a composition.

The need for each of the FDM CAs in the composition to be invertible puts additional restrictions on their form. In order to be sure that the FDM CA is invertible, the set T which is used to find the acting set of each FDM CA must contain all states that the function f can change. The functions $f_1, f_2, \ldots, f_n$ must also be permutations (one-to-one mappings). We discuss the key generation algorithm in more detail in Section 4.1.

Once the component FDM CAs are generated, the public key is determined by sequentially applying $C_1, C_2, \ldots, C_n$ to each possible neighbourhood (using the neighbourhood as the starting configuration). The final state of the cell is recorded, and the public key is this mapping of neighbourhoods to states. The private key is not calculated explicitly; the CAs $C_1^{-1}, C_2^{-1}, \ldots, C_n^{-1}$ are simply applied sequentially for decryption. The message is encoded in a d dimensional grid and is evolved for a fixed number of iterations of the public key to produce the ciphertext. The ciphertext and number of iterations are sent as the encrypted message.

4.1 The Key Generation Algorithm

Our key generation scheme is given in Algorithm 1. We should note that in this algorithm, the `random_element` function returns a random element from a given set, the `random` function returns a floating point number between 0.0 and 1.0, the `random_permutation` function returns a random permutation mapping of a given set, and the `random_binary` function returns a random binary string of a given length. Also, the `get_all_possible_neighbourhoods` function returns all possible neighbourhoods given a state set S and neighbourhood N.

Initially T is a set $T \subseteq S$ of two random elements of S. The FDM CAs in the composition are then constructed in order from C_1 to C_n. During the generation

Input: State space S, Neighbourhood N, Number of FDM CA n, $0 < p, q < 1$
Output: Set of reversible FDM CAs $C_1, C_2, \ldots, C_n$

```
Initialization
T ⟵ ∅
T.add (random_element (S))
T.add (random_element (S−T))
all_possible_neighbourhoods ⟵ get_all_possible_neighbourhoods (S,N)

for i ← 1 to n do
    The following code determines f_i
    if random () < p and T ≠ S then
        T.add (random_element (S−T))
    end
    f_i ⟵ random_permutation (T)

    The following code determines A_i
    binary_string ⟵ random_binary (|N|)
    A_i ⟵ ∅
    for neighbourhood ∈ all_possible_neighbourhoods do
        unchanging_neighbourhood ⟵ True
        for j ← 1 to |N| do
            if neighbourhood [j] ∈ T then
                unchanging_neighbourhood ⟵ False
                if binary_string [j] = 1 then
                    A_i.add (neighbourhood)
                    break
                end
            end
        end
        if unchanging_neighbourhood = True and random () < q then
            A_i.add (neighbourhood)
        end
    end
    C_i ⟵ { S,N,A_i,f_i }
end
```

Algorithm 1: The public-key generation algorithm, discussed in Section 4.1.

of each C_i, with probability p a new element from S is added to the set T, and otherwise T stays the same. The function f is chosen for each FDM CA to be a random permutation of the set T, and f applies the identity map to states in $S - T$. We should note that one should choose n to be sufficiently large so that $T = S$ at some point (i.e. no states are left completely unchanged), but we will discuss this more later.

The only remaining task is to select the acting set. For each FDM CA in the composition, a random binary string of length $|N|$ is chosen. Every possible neighbourhood is then considered as a candidate element of the acting set. If the candidate neighbourhood has a state which is an element of T and is also in a position corresponding to a '1' of the binary string, then it is added to the acting set. Also, if the neighbourhood contains only states which are not in T, then the neighbourhood is added to the acting set with probability q. For example, consider the case where $S = \{a, b, c\}$, $N = \{-1, 1\}$, $T = \{a, b\}$, and the random binary string is 01. Then the neighbourhood ca is a member of the acting set while ac is not, and cc is a member of the acting set with probability q. Note that if the neighbourhood N contains the zero element, then clearly the case where the neighbourhood is added to the acting set with probability q is irrelevant since not even the state of the cell can change.

We now discuss the correctness of this algorithm, and begin by showing that the condition for constant neighbourhood size during composition holds. Assume we are attempting to determine the acting set of the i^{th} FDM CA in the composition, A_i, and let us first consider neighbourhoods which have at least one state in T. If a neighbourhood is in A_i, then at least one element of T occurring in the neighborhood corresponds to a '1' in the binary string. Since $C_i \circ C_{i-1} \circ \cdots \circ C_1$ is T-invariant (states in T are mapped to states in T), the neighbourhood will certainly be mapped to a neighbourhood in A_i. On the other hand, if a neighbourhood is not in A_i, then all occurrences of states in T correspond to '0' elements of the binary string. This neighbourhood is mapped to a neighbourhood where states in T also correspond to '0' elements of the binary string, and hence it is mapped to a neighbourhood which is not in A_i. Finally, if we consider a neighbourhood which contains no elements of T, then clearly the condition of Proposition 2 is satisfied regardless of whether the neighbourhood is in A_i, or whether N contains the zero element, since the neighbourhood must map to itself.

It remains to show that the condition for FDM CA reversibility holds for each C_i. Rather conveniently, the previous conditions actually allow (or demand) that A_i is also the acting set of the inverse FDM CA. Since any addition to the set T during the construction of each FDM CA happens before we choose A_i, we are guaranteed that elements of A_i will be mapped to elements of A_i, and elements not in A_i will not be mapped to A_i. So the condition from Proposition 4 also holds. Note that we could not be sure of this if A_i was constructed with some T that did not correspond with the states that f_i changes.

4.2 Security Concerns and Practical Considerations

Since the FDM CA compositions follow a specific form and are not general two-dimensional RCA, we cannot directly use Kari's result [9] to justify the security of the system, and hence the security of this cryptosystem is largely unknown to us. However, we do not believe that straightforward brute force attacks will work. If one attempted to guess at a composition of FDM CAs which resulted in the same public key, there are many choices for each CA and there are $n!$ ways to arrange the rules, since n is the number of rules in the composition. One could also attempt to keep track of all global inputs and outputs for a fixed grid size in order to invert the composed CA. In this case the number of possible global configurations is $|S|^{(g)}$ where g is the number of grid cells, so as long as the grid (the message) is relatively large this method will not work.

We also do not believe that the private key $C_1^{-1} \circ C_2^{-1} \circ \cdots \circ C_n^{-1}$ can be guessed very easily. Although we do not calculate it explicitly, this CA must have a fairly large neighbourhood because for each composition in the sequence, the condition from Proposition 2 does not hold in general. Each time T changes during the generation of the FDM CA, the inverse automaton's neighbourhood size may increase, and this can happen at most $|S| - 2$ times. So there is a computable upper bound for the neighbourhood size of the inverse, given $C_1, C_2, \ldots, C_n$, but for reasonably large S and $d > 1$ this probably does not pose a security threat.

A user must choose the parameters of our algorithm with some care in order to prevent these brute force attacks and also to be able to encrypt and decrypt within a reasonable amount of time on a normal computer. One such setup might be $N = \{(0,1),(1,0)\}$ (the top and the right neighbours), $|S| \approx 25$, grid size $g \approx 500$, number of FDM CA in the composition $n \approx 100$, $p = q = 0.5$, and number of iterations ≈ 100. These sizes can probably be increased significantly if the algorithm were implemented on specialized parallel hardware (especially the grid size and number of iterations). We should note that the expected number of CA needed in the composition to just achieve $T = S$ is $(|S| - 2) \cdot 1/p$, and so n should be chosen so that it is significantly larger than this quantity. If n is too small, then the composition will only change states in T, and all elements in $S - T$ that occur in the original message will occur in the same places in the ciphertext.

One security issue related to the last point is that with our key generation algorithm as written, it is very easy for an attacker to determine which state was last added to T. The public key will map this state to some other state regardless of the neighbourhood. Not much can be immediately done with this information, but perhaps it could be a starting point for a clever cryptanalytic algorithm to find each of the FDM CA in the composition in backwards order.

5 Conclusion and Future Work

We presented conditions which guarantee that compositions of fixed-domain marker cellular automata have the same neighbourhood as each of the individual components. We showed that, under certain technical assumptions, an FDM

CA has a unique inverse with a given neighbourhood. We used these results to design, present, and show the correctness of a working key generation algorithm for a public-key cryptosystem originally conceived by Kari [8]. We also provided some preliminary cryptanalysis and gave some practical implementation notes.

This work provides several avenues for further research. We have given perhaps a more manageable definition of marker cellular automata, which could facilitate or help with additional theoretical development in related areas. The security of the cryptosystem presented in this work is currently unknown, and serious cryptanalysis is needed before more can be said in this regard. Also, there may be some alternate or more general way to choose the acting sets of each CA in the composition, which could result in a more secure or efficient system. If the cryptosystem does not break easily then it would make sense to try to design an optimal hardware implementation and to do a corresponding feasibility analysis for real-world applications.

References

1. Amoroso, S., Patt, Y.: Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tesselation Structures. J. Comput. System Sci. 6, 448–464 (1972)
2. Anghelescu, P., Ionita, S., Sofron, E.: Block Encryption Using Hybrid Additive Cellular Automata. 7th Int. Con. on Hybrid Intelligent Systems, 132–137 (2007)
3. Ashley, J.: Marker automorphisms of the one-sided d-shift, Ergodic Theory Dynam. Systems 10, No. 2, 247-262 (1990)
4. Charbouillot, S., Perez, A., and Fronte, D.: A Programmable Hardware Cellular Automaton : Example of Data Flow Transformation. 13th IEEE International Conference on Electronics, Circuits and Systems. 1232–1235 (2006)
5. Franti, E., Slav, C., Balan, T., and Dascalu, M.: Design of cellular automata hardware for cryptographic applications. CAS 2004 Int. Semiconductor Conference Vol. 2, 463–466 (2004)
6. Gutowitz, H.: Cryptography with Dynamical Systems. In: Cellular Automata and Cooperative Phenomena, Eds: E. Goles and N. Boccara, 237–274. Kluwer Academic Press (1993)
7. Gutowitz, H.: Method and Apparatus for Encryption, Decryption, and Authentication using Dynamical Systems, U.S. Patent 5365589 (1994)
8. Kari, J.: Cryptosystems based on reversible cellular automata. Manuscript (1992)
9. Kari, J.: Reversibility and Surjectivity Problems of Cellular Automata. J. Comput. System Sci. 48, 149–182 (1994)
10. Seredynski, M., and Bouvry, P.: Block cipher based on reversible cellular automata. New Gen. Comput. 23, 3 245–258 (2005)
11. Srebrny, M., and Such, P.: Encryption using two-dimensional cellular automata with applications. In Artificial intelligence and Security in Computing Systems, 203–215. Kluwer Academic Publishers (2003)
12. Tao, R. and Chen, S.: On finite automaton public-key cryptosystem. Theoretical Computer Science 226 No.1-2, 143-172 (1999)
13. Wolfram, S.: Random sequence generation by cellular automata. Advances in Applied Mathematics 7 No.2, 163–169 (1986)
14. Zheng, Y., and Imai, H.: A cellular automaton based fast one-way hash function suitable for hardware implementation. In Public Key Cryptography, number 1431 in Lecture Notes in Computer Science, pp.217-234, 1998.