ON ENCRYPTION SYSTEMS REALIZED BY FINITE TRANSDUCERS

A. Demers

C. Kelement†

B. Reusch††

TR 76-291

Department of Computer Science
Cornell University
Ithaca, New York 14853

†On leave from Ithaca College, Ithaca, New York 14850
††On leave from Universitat Dortmund, Abteilung Informatik, 46 Dortmund 50, Postfach 500500, West Germany

ON ENCRYPTION SYSTEMS REALIZED BY FINITE TRANSDUCERS

A. Demers
C. Kelemen
B. Reusch

In [S & P] Sardinas and Patterson give a necessary and sufficient condition on the set of code words for the unique decomposition of coded messages into individual code words. For example, if 'give', 'given', 'one', and 'none' are all code words then the coded message 'givenone' is ambiguous because it could be decomposed into 'given one' or 'give none'. In [B], Brückner gives a condition that guarantees an unambiguous word code can be decoded by a finite transducer and shows how to construct the decoding transducer. The types of encodings considered in both [S & P] and [B] can be realized by very simple deterministic finite transducers. However, there is a wide variety of enciphering techniques realizable by finite transducers but not covered by the results of [S & P] and [B]. For example the Hagelin machine used extensively for military communication during World War II [K] and the proposed Data Encryption Standard of the National Bureau of Standards [NBS] fall into this category. Both these encryption systems can be handled by our results.

Encryption is an important technique to be considered in any data or communication security system [NBS], [Turn], [Hoffman]. It is the process of transforming information into an unintelligible form called a cipher. The original information is called the plain-text, the transformation process is called enciphering and the transformed plain-text is called the cipher-text. The process of reversing the enciphering transformation to obtain plain-text from the cipher-text is called deciphering when total information about the enciphering transformation is known. Determining plain-text from cipher-text with less than total information about the enciphering process is called cryptanalysis. Authorized recipients of cipher-text decipher it, unauthorized recipients are forced to employ the methods of cryptanalysis to obtain the plain-text. A good encryption system makes enciphering and deciphering easy and inexpensive but cryptanalysis extremely hard and costly. While it is essential that a cipher be unambiguous (i.e. uniquely decipherable) and as a practical matter decipherable by a deterministic device, it is not necessary that the enciphering device be deterministic. In fact, probabalistic choices in the enciphering device might be used to increase the difficulty of cryptanalysis. In section 2 we show that, for encipherers realized by a finite transducer, ambiguity, single-valuedness, and equivalence are all decidable properties. Section 3 takes up the problem of deterministically deciphering the output of a finite transducer encipherer. It is shown that a deterministic pebble machine is sufficient to decipher the output of a finite transducer encipherer and that the existence of a deterministic finite transducer decipherer is decidable (the proof provides an effective construction when existence is shown). In section 4, it is shown that for encipherers realized by pushdown transducers ambiguity is undecidable.

1. Background and Notation

In this section we recall some basic ideas from the theory of translation. Our presentation basically follows Aho and Ullman [A & U], but our notation differs slightly from theirs. We assume familiarity with the theory of regular languages and finite-state machines, and we assume some knowledge of the theory of context-free languages. Both topics are covered in detail in [A & U].

Let Σ and Δ be finite alphabets. A translation from Σ^* to Δ^* is a set $T \subseteq (\Sigma^* \times \Delta^*)$. Σ and Δ are the input and output alphabets, respectively. The domain of T, denoted dom T, is the set $\{x \in \Sigma^* | (\exists y \in \Delta^*)\ (x,y) \in T\}$. Similarly, range T is $\{y \in \Delta^* | (\exists x \in \Sigma^*)\ (x,y) \in T\}$. T defines a mapping from Σ^* to subsets of Δ^* in the obvious way: for $x \in \Sigma^*$, $T(x) = \{y \in \Delta^* | (x,y) \in T\}$. It is often useful to apply this mapping to sets of strings: for $S \subseteq \Sigma^*$,

$$T(S) = \bigcup_{x \in S} T(x).$$

Given a translation T from Σ^* to Δ^*, we define T^{-1}, the <u>inverse of T</u>, as $T^{-1} = \{(y,x) \mid (x,y) \in T\}$. Viewed as mappings, T and T^{-1} are not truly inverses, since, for example, $T^{-1}(T(x)) = \{x' \in \Sigma^* \mid T(x') = T(x)\}$ is a set which contains x, perhaps properly.

A number of other properties of translations are defined in analogy to mappings. A translation $T \subseteq \Sigma^* \times \Delta^*$ is

<u>total</u> if for every $x \in \Sigma^*$, $T(x) \neq \phi$,
<u>single-valued</u> if for every $x \in \Sigma^*$, $\#(T(x)) \leq 1$ and
<u>unambiguous</u> if T^{-1} is single-valued.

Our interest lies primarily with translations which can be performed by a processor with a finite memory. Formally, we define a (<u>nondeterministic</u>) <u>finite-state transducer</u> (ft) as a system $M = (Q,\Sigma,\Delta,\delta,q_0,F)$, where

i) Q is a finite set of <u>states</u>,
ii) Σ and Δ are finite <u>input</u> and <u>output</u> alphabets, respectively,
iii) $q_0 \in Q$ is the <u>initial state</u>,
iv) $F \subseteq Q$ is the set of <u>final states</u>, and
v) δ, the <u>transition function</u> of M, maps $Q \times (\Sigma \cup \{e\})$

to finite subsets of $Q \times \Delta^*$.

The interpretation of the transition function is similar to that for finite-state acceptors: if $(q,w) \in \delta(p,a)$ then from state p, M can read input a, emit output w, and transfer to state q. We write

a) $p \to (a/w)q \qquad a \in \Sigma \cup \{e\}, (q,w) \in \delta(p,a)$,

to represent a single move of M. A sequence of zero or more moves can be built up in the obvious way:

b) $p \overset{*}{\rightarrow} (e/e)p$ for all $p \in Q$

c) if $p \overset{*}{\rightarrow} (u/w)q$ and $q \overset{*}{\rightarrow} (v/x)r$, then $p \overset{*}{\rightarrow} (uv/wx)r$.

M defines a translation $\tau(M) \subseteq \Sigma^* \times \Delta^*$ by $(x,y) \in \tau(M)$ iff $q_0 \overset{*}{\rightarrow} (x/y)q_f$ for some $q_f \in F$. Notationally, we identify M and $\tau(M)$, writing

$$
\begin{aligned}
M(x) &= \tau(M)(x),\\
\text{dom } M &= \text{dom } \tau(M),\\
\text{range } M &= \text{range } \tau(M),\\
M \text{ is single-valued} &\equiv \tau(M) \text{ is single-valued,}
\end{aligned}
$$

and so forth. T is a <u>regular translation</u> if there exists a ft M such that $T = \tau(M)$.

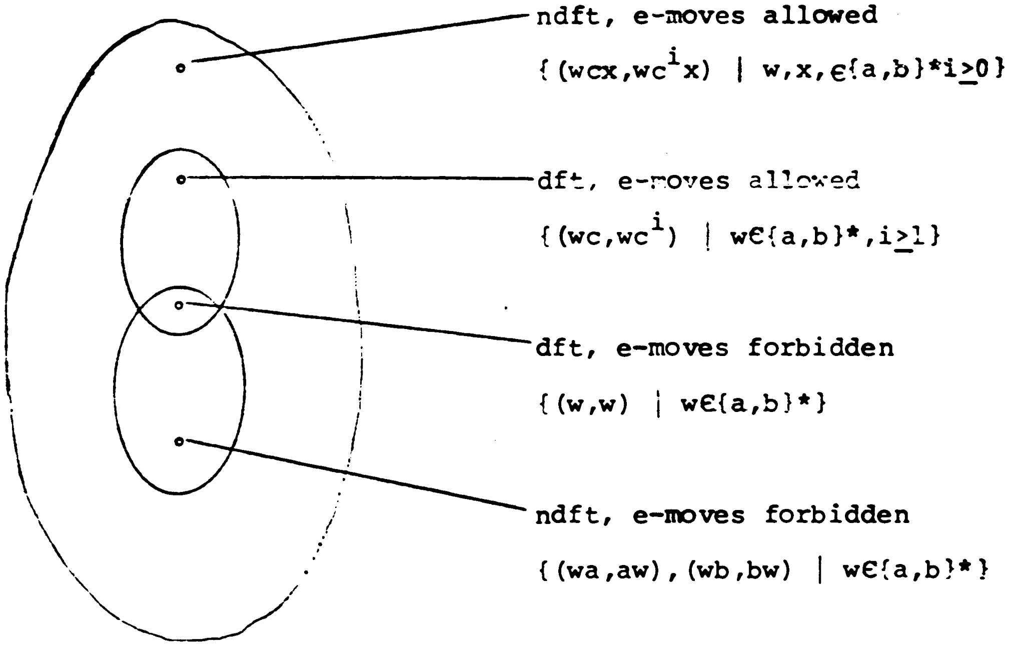
We usually represent a ft by its <u>transition graph</u>, an unordered edge-labeled directed graph. Nodes of the graph represent states of the transducer. An edge labeled x/y connects nodes p and q iff $p \rightarrow (x/y)q$.

The following facts are well known. Proofs can be found in Aho & Ullman [A & U].

<u>Theorem 1.1</u>: If T is a regular translation, then T^{-1} is a regular translation. Moreover, if $T = \tau(M)$, then we can effectively construct M^{-1} such that $T^{-1} = \tau(M^{-1})$. □

<u>Theorem 1.2</u>: Let T be a regular translation, R a regular set, and L a context-free language. Then

(a) dom T and range T are effectively regular,
(b) $T(R)$ is effectively regular, and
(c) $T(L)$ is effectively context-free. □


ndft, e-moves allowed
$\{(wcx, wc^{i}x) \mid w, x, \in \{a,b\}^{*} i \geq 0\}$
dft, e-moves allowed
$\{(wc, wc^{i}) \mid w \in \{a,b\}^{*}, i \geq 1\}$
dft, e-moves forbidden
$\{(w,w) \mid w \in \{a,b\}^{*}\}$
ndft, e-moves forbidden
$\{(wa,aw), (wb,bw) \mid w \in \{a,b\}^{*}\}$

<u>Figure 1.1</u>

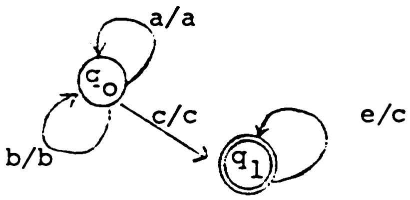


Figure 1.2

The deterministic version of finite transducer can be defined in at least two ways which seem natural. A ft $M = (Q,\Sigma,\Delta,d,q_0,F)$ is

(a) <u>deterministic</u> (a <u>dft</u>) if for all $q \in Q$ and $a \in \Sigma$, $\#(d(q,a)) + \#(d(q,e)) \leq 1$.

(b) a <u>generalized</u> <u>sequential</u> <u>machine</u> (gsm) if M is a dft with no moves on e input - i.e., for any $q \in Q$, $d(q,e) = \phi$

The reader familiar with the theory of finite automata will recall that nondeterminism and e-moves do not add to the power of finite acceptors. That is, to any finite state acceptor there is an equivalent deterministic one without e-moves. For transducers this is not the case. Both nondeterminism and e-moves contribute to the power of such devices, resulting in the situation depicted in Figure 1.1. The figure gives a translation for each region of the Venn diagram; it is easy to verify that the translations have the properties claimed.

It is interesting to note that while $\tau(M)$ is single-valued for any gsm M, this need not be true of a dft.

<u>Example 1.1</u>: Consider the dft M of Figure 1.2. The domain of M is $\{a,b\}^*c$; any input wc from this set causes M to output wc and reach its final state q_1. The e-move from q_1 to itself can then be performed any number of times, outputting an additional c each time. Thus, $T(wc) = \{wc^i | i \geq 1\}$, an infinite set. □

Deciding whether a given ft is single-valued is important in dealing with ciphers. This question is treated at length in Section 2.

2. Finite Encipherers

In this section we consider enciphering by finite-state machines. In particular, we show that it is decidable whether a finite-state translation has a well-defined inverse - i.e., whether a cipher is decipherable.

For enciphering and deciphering we shall find it convenient to assume the existence of endmarkers. These can be included in translations in a natural way:

Definition 2.1: Let $T \subseteq \Sigma^* \times \Delta^*$ be a translation, and let $\$ \notin \Sigma \cup \Delta$. Then $T\$$ is the translation $\{(x\$,y\$) \mid (x,y) \in T\}$.
We can now include endmarkers explicitly in the definition of a cipher:

Definition 2.2: A translation $T \subseteq \Sigma^* \times \Delta^*$ is a finite transducer cipher (ft-cipher) if $T\$$ is $\tau(M)$ for some ft M, where $\$ \notin \Sigma \cup \Delta$. T is a deterministic ft cipher if $T\$$ is $\tau(M)$ for some dft M.

Many important properties of translations are preserved by addition or removal of endmarkers. In particular, we have the following:

Lemma 2.1: For translation $T \subseteq \Sigma^* \times \Delta^*$ and $\$ \notin \Sigma \cup \Delta$,

(a) $(T\$)^{-1} = (T^{-1})\$$.

(b) $T\$$ is single-valued iff T is.

(c) $T\$$ is unambiguous iff T is.

Proof: Obvious. □

When nondeterministic ft's are used as enciphering machines, the addition of endmarkers gives us no additional power. Formally,

Theorem 2.1: $T\$$ is a regular translation iff T is a regular translation.

Proof: Follows easily from the fact that T is regular iff T is characterized by some regular language (see [AU]). The details are omitted. □

If we restrict our attention to deterministic devices, however, the above theorem fails.

Example 2.1: Let T be the translation $\{(w,w2) \mid w \in \{0,1\}^*\}$. It is easily seen that if M is a deterministic ft, then $M(w)$ must be a prefix of $M(wx)$ for any w and wx in dom M. Choosing $x \neq e$, we obtain $T(w) = w2$, which is not a prefix of $wx2 = T(wx)$; thus T cannot be $\tau(M)$ for any dft M.

Nevertheless, T is a dft cipher. The addition of endmarkers makes the domain of the translation prefix-free, thereby enabling it to be performed by a dft. The reader can verify that the machine M' of Figure 2.1 does in fact compute $T\$$.

Another effect that endmarkers have on deterministic translations can be seen in the following

Lemma 2.2: If T is a dft cipher then T is single-valued.

Proof: If $T \subseteq \Sigma^* \times \Delta^*$ is a dft cipher, by definition there exists a dft $M = (Q, \Sigma \cup \{\$\}, \Delta \cup \{\$\}, d, q_0, F)$ such that $T\$ = \tau(M)$ (where $\$ \notin \Sigma \cup \Delta$). Let $x \in$ dom T (where $x\$ \in$ dom M), and consider the shortest accepting computation of M on input $x\$$:

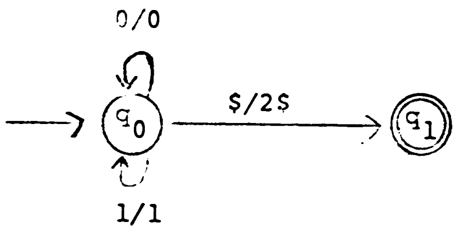


Figure 2.1

$$q_0 \overset{*}{\vdash} (x\$/y\$)\ q_f \qquad \text{for some } q_f \in F.$$

Since M is deterministic, the above computation is a prefix of every accepting computation of M on input x$; thus y$ is a prefix of every word in M(x$). Since range M $\subseteq \Delta^*\$$ is prefix-free, this implies M is a single-valued. □

If a cipher is to be of any practical use, it must be possible at least in principle to decipher it; that is, a useful cipher must be unambiguous. The main result of this section is an efficient algorithm which decides whether an arbitrary ft is ambiguous.

We have chosen nondeterministic ft's as our enciphering machine model. This was done partly for generality - most of our results apply to nondeterministic as well as deterministic ft's. But it is also important to note that a nondeterministic machine can often be simulated by a machine which makes probabilistic choices. Such a simulation can result in a cipher which is unambiguous but multi-valued, and is therefore more difficult to "break" by cryptanalysis than a single-valued cipher would be.

Probabilistic simulation of a ndft is fairly straightforward if the encipherer is allowed to make more than one pass over the entire plain text input. Even such a simple machine as probabilistic 1 - pebble transducer (see Section 3) can simulate an arbitrary ndft, and a probabilistic 2-way pushdown transducer, can perform the simulation in linear time. The constructions are like the analogous constructions for deciphering machines, and we omit them.

There is a natural probablistic simulation method which requires only a single left-to-right scan of the input. Let us call this technique <u>direct probabilistic simulation</u>:

> Given an ndft M and input x, the simulator behaves like M(x) except that, whenever a nondeterministic choice of M is encountered, the simulator picks one of the possible moves at random, ignoring all other possible moves of M.

A computation of the simulator on input x corresponds to a computation of M chosen at random from among the many possible computations of M on input x. Since not all computations of a ndft lead to acceptance, it follows that this naive simulation technique is <u>unsafe</u> - the simulator may fail to produce a translation for some valid input.

<u>Example 2.2</u>: We construct a ndft for which direct probablistic simulation is unsafe. Let $\Sigma = \{a,b,c\}$, $\Delta = \{a,b,c,d,e,f\}$. For $w \in \Sigma^*$, let $\bar{w}$ denote the result of replacing a by d, b by e, and c by f in w. Let $T = \{(wa,wa),(wb,wb) | w \in \Sigma^*\} \cup \{(wb,\bar{w}b),(wc,\bar{w}c) | w \in \Sigma^*\}$. Clearly, dom $T = \Sigma^+$. T\$ is performed by the ndft M given in Figure <u>2.2</u>. The only nondeterministic choice in M occurs from state q_0, where without reading an input a probabilistic simulator chooses to go either to state q_1 or q_2. If it chooses q_1 and the input string ends with c, then the final state q_5 cannot be reached and the

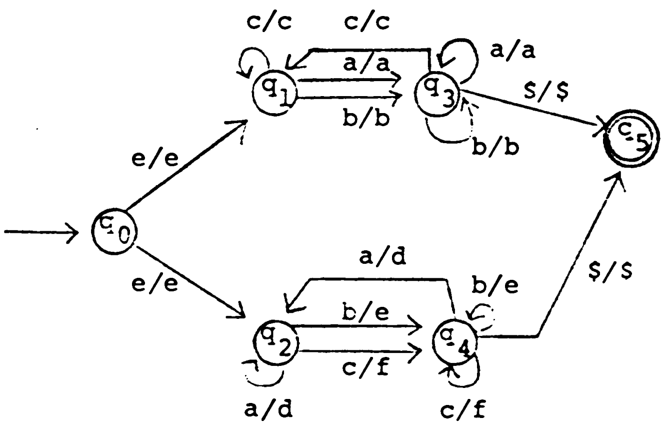


Figure 2.2

simulation fails. Similarly, if q_2 is chosen and the input ends with a, again the simulation fails.

A simple characterization of the ndft's for which direct probablistic simulation is safe comes from the following.

Definition 2.3: Let $M = (Q,\Sigma,\Delta,d,q_0,F)$ be a ft. Two states p and q are input-equivalent if, for all $x \in \Sigma^*$,

$$(\exists y \in \Delta^*)(\exists p_f \in F) \quad p \overset{*}{\rightarrow} (x/y) p_f$$

if and only if

$$(\exists z \in \Delta^*)(\exists q_f \in F) \quad q \overset{*}{\rightarrow} (x/z) q_f.$$

That is, two states are input equivalent if the languages recognized, starting from those states, are equal. Now the desired safety criterion is simply that nondeterministic choices are made only between input-equivalent states.

Lemma 2.3: Direct probabilistic simulation of a ndft $M = (Q,\Sigma,\Delta,\delta,q_0,F)$ is safe if and only if

(a) For any $p \in Q$, $a \in \Sigma$, if $d(p,a)$ contains (q,u) and (r,v) then q and r are input-equivalent, and

(b) for any $p \in Q$, if $d(p,e)$ contains (q,u) then p and q are input-equivalent.

Proof: Straightforward. □

We now present the main result of this section, that ambiguity of a ndft is decidable. This result is related to, and implies, decidability of ambiguity of a regular grammar; however, the notions of ambiguity for grammars and ciphers are quite different.

A few preliminary remarks are needed.

First, recall from Theorem 1.1 that for any ft M we can effectively construct a (nd)ft M^{-1} such that $\tau(M^{-1}) = (\tau(M))^{-1}$. Since by Definition M is unambiguous if and only if M^{-1} is single-valued, it suffices for us to show that single-valuedness is decidable, and this is what we shall do.

For this discussion we choose fixed input and output alphabets Σ and Δ, respectively. Let Pal denote the set of palindromes over Δ with centermarker $\# \notin \Delta$; that is,

$$\mathrm{Pal} = \{w \# w^R | w \in \Delta^*\}.$$

Hopcroft [H1] has shown that it is decidable whether an arbitrary context-free grammar generates a subset of Pal, and our proof is based on this fact.

Let M_1 and M_2 be (nd)ft's, where

$M_i = (Q_i, \Sigma, \Delta, \delta_i, q_{0i}, F_i) \quad i = 1,2.$

Note the machines have the same input and output alphabets.

Definition 2.4: The output pair language for M_1 and M_2 is the set

$$\mathrm{OPL}(M_1,M_2) = \{x \# y^R | (\exists w)(w,x) \in \tau(M_1) \wedge (w,y) \in \tau(M_2)\}.$$

Definition 2.5: The output pair grammar for M_1 and M_2 is the cfg

$$\mathrm{OPG}(M_1,M_2) = (N, \Delta \cup \{\#\}, P, S),$$

where

$N = Q_1 \times Q_2 = \{[p,q] \mid p \in Q_1,\ q \in Q_2\}$

$S = [q_{01}, q_{02}]$

P contains

$[p,q] \to u[p',q]$ if $(p',u) \in \delta_1(p,e)$

$[p,q] \to [p,q']v^R$ if $(q',v) \in \delta_2(q,e)$

$[p,q] \to u[p',q']v^R$ if $(\exists a \in \Sigma)$
$(p',u) \in \delta_1(p,a) \wedge$
$(q',v) \in \delta_2(q,a)$

$[p,q] \to \#$ if $p \in F_1,\ q \in F_2$.

Naturally, we have the following

Lemma 2.4: $L(OPG(M_1,M_2)) = OPL(M_1,M_2)$.

Proof: Routine. □

Now the question of single-valuedness of a ndft can be reduced to equivalence to Pal.

Theorem 2.2: It is decidable whether a ndft is single-valued.

Proof: By definition, a ndft M is single-valued iff $(\forall x)[[(x,y) \in \tau(M) \wedge (x,z) \in \tau(M)] \Rightarrow [y=z]]$. But (x,y) and (x,z) are in $\tau(M)$ for some x if and only if $y\#z^R \in OPL(M,M)$ Thus, M is single-valued iff $[y\#z^R \in OPL(M,M)] \Rightarrow [y=z]$; i.e., $OPL(M,M) \subseteq Pal$. To decide single-valuedness of M, it suffices to construct $G = OPG(M,M)$ and then use the algorithm of Hopcroft [H1] to decide whether $L(G) = Pal$. □

Corollary: It is decidable whether a ndft is unambiguous.

Remark: Hopcroft's algorithm decides for an arbitrary cfg G whether $L(G) \subseteq Pal$, and may require exponential time. However, output pair grammars are not arbitrary, but are linear grammars. For these grammars, the algorithm can be modified to run in time which is linear in the size (i.e., the sum of the lengths of the production right sides) of the grammar. Using a similar definition for the size of a ndft, the sizes of M and M^{-1} are linearly related, and the size of OPG(M,M) is at most quadratic in the size of M. Thus, our entire algorithm can be made to run in time $O(n^2)$.

To close this section, it is worth pointing out that the equivalence problem for single-valued ft's is solvable by a fairly obvious algorithm. For ft's M_1 and M_2, let $M_1 \| M_2$ be the nondeterministic machine which chooses initially to behave like either M_1 or M_2, so that $\tau(M_1 \| M_2) = \tau(M_1) \cup \tau(M_2)$. Then we have the following

Lemma 2.5: Single-valued ft's M_1 and M_2 are equivalent if and only if

(a) $\mathrm{dom}(M_1) = \mathrm{dom}(M_2)$, and

(b) $M_1 \| M_2$ is single-valued.

Proof: Straightforward. $\square$

If our size measure for transducers is such that we can enumerate the ft's in order of increasing size, then we could in principle use the above result to construct a minimal transducer equivalent to a given one.

Lemma 2.5 is in contrast to the situation with arbitrary (multi-valued, ambiguous) ft's, for which the equivalence problem is unsolvable [G]. Hopcroft [H2] has observed that the equivalence problem for deterministic ft's can be reduced to the equivalence problem for one-way, two-tape finite automata, which is known to be solvable [Bird]. However, the decidability results for deterministic and single-valued ft's are independent - neither result implies the other.

3. Deciphering the Output of ft Encipherers

It is not difficult to construct unambiguous ciphers realized by deterministic finite transducers that cannot be deciphered by a deterministic finite transducer. Two questions naturally arise. What sort of finite state device is sufficient to deterministically decipher the output of an unambiguous ft encipherer, and is it decidable whether a deterministic finite transducer decipherer exists for a given ft encipherer? In this section we show that the output of any unambiguous ft encipherer can be deciphered by a deterministic pebble transducer. We also show that it is decidable whether a deterministic ft decipherer exists for a given ft encipherer. The proof provides an effective construction for a dft decipherer whenever one exists.

Definition 3.1: A deterministic pebble transducer is a deterministic finite transducer enhanced with the ability to read the input tape in both directions and to nondestructively place, remove, and sense a single pebble on any input square.

Theorem 3.1: Let $M = (\Sigma, Q, \Delta, \delta, q_0, F)$ be an unambiguous ft encipherer. Then a decipherer for M can be realized by a deterministic pebble transducer.

Proof: Label each transition of M (i.e., each edge in the transition diagram of M) by a unique integer, so the edges are labeled $1,2,\ldots,n$. Let the i^{th} transition be $p \to (a/\alpha)q$. We could construct the ndft $M_i = (Q \cup \{p_i\}, \Sigma, \Delta, \delta_i, p_i, F)$ such that $\delta_i(p_i, e) = \{(q,\alpha)\}$ and $\delta_i = \delta$ on $Q \times \Sigma^*$. For all $1 \leq i \leq n$, Range M_i is a regular set (Theorem 1.2). In fact

$$\text{Range } M_i = \{\alpha\beta \mid (\exists w)\ \delta(q,w) \text{ contains } (r,\beta) \text{ for some } r \in F\}$$

is the set of outputs of accepting computations of M starting with some occurence of the i^{th} transition.

By dropping its pebble, scanning to the end of its input tape and then moving back to the pebble, a deterministic pebble transducer in configuration (w,p,x) can compute $\{i \mid x \in \text{Range } M_i\}$ by

$$(*) \quad (w,p,x) \vdash (w\Diamond,p,x) \vdash^* (w\Diamond x,q,e) \vdash^* (w,r,\Diamond x) \vdash (w,s,x)$$

where the state s encodes the desired information.

The pebble machine can now simulate some computation of M, using a "subroutine" of the form (*) as an oracle. That is, in the pebble machine,

$$(w\Diamond,p,x) \vdash^* (wx_1\Diamond,q,x_2)$$

with output a, where $x = x_1x_2$ and $p \to (a/x_1)q$ is the transition of least index i which starts from state p and such that $x \in \text{Range } M_i$. This transition can be determined by using the subroutine (*). □

Note that we did not need the full power of a pebble machine in the above argument. We do not use the ability to read backwards except to be able to backspace the input tape to the pebble square.

Furthermore, since a pebble machine is weaker than a 1-counter machine it follows that deterministic 1-counter machines are powerful enough to decipher any message enciphered by an ft. With the full power of a pushdown transducer the deciphering can be done in linear time.

In practice, however, one wants a decipherer to be a finite, deterministic machine. Thus, it is of some interest that, given an (unambiguous) cipher; we can decide whether a deterministic ft decipherer exists. Our proof is a constructive one; we exhibit a dft decipherer if one exists.

We continue working with decipherers (rather than encipherers) assuming single-valuedness of the decipherers, (or equivalently, unambiguity of the corresponding encipherers). We give two properties which together characterize those nondeterministic finite state translations which can be performed deterministically. We then show that these properties are decidable, giving the desired result.

<u>Properties 3.1</u>: Let M be a single-valued (nd)ft. We claim the following properties are necessary and sufficient conditions for $\tau(M)$ to be deterministic.

(a) there exists $k \geq 0$ such that, for any $w \in \Sigma^*$; $x,y \in \Delta^*$; and $p,q \in Q$,

$$q_0 \xrightarrow[M]{*} (w/x)p, \quad q_0 \xrightarrow[M]{*} (w/y)q \text{ implies } ||x|-|y|| \leq k.$$

(b) there exists $\ell \geq 0$ such that, for any $w \in \Sigma^*$, $x,y,z \in \Delta^*$, $a \neq b \in \Delta$, and $p,q \in Q$

$$q_0 \xrightarrow[M]{*} (w/xay)p, \quad q_0 \xrightarrow[M]{*} (w/xbz)q \text{ implies } |aybz| \leq \ell$$

To prove these claims we need the following technical lemma, which says partial outputs of a single-valued ndft can't be "much longer" than inputs.

<u>Lemma 3.1</u>: Let M be (nd)ft. Let n = # states of M, d = max # symbols output in any single move of M. If there exist $p,q \in Q$, $w \in \Sigma^*$, $x \in \Delta^*$ s.t. $|x| > (|w| + 1)(n-1)d + |w|d$ and $p \xrightarrow[M]{*} (w/x)q$ then M has a cycle of e-moves with nonempty output (hence can't be single-vauled.)

<u>Proof</u>: Simple counting argument. □

<u>Lemma 3.2</u>: If $\tau(M)$ is deterministic then M has property 3.1(a)

<u>Proof</u>: Let $M = (Q,\Sigma,\Delta,\delta,q_0,F)$; Let $M' = (Q',\Sigma \cup \{\$\}, \Delta \cup \{\$\},\delta',q'_0,F')$ be a deterministic ft such that $\tau(M') = \{(x\$,y\$) \mid (x,y) \in \tau(M)\}$. M' must exist by definition of a deterministic translation. Let

n = #Q = # states of M
n' = #Q' = # states of M'
d = max # symbols emitted by any single move of M
d' = max # symbols emitted by any single more of M'.

Assume condition (a) doesn't hold; then $k \geq 0$, w,x,y,p,q s.t.

(*) $q_0 \xrightarrow[M]{} (w/x)p$, $q_0 \xrightarrow[M]{} (w/y)q$, and $|y|-|x| > k$.

We will show by Lemma 3.1 that M' cannot be single-valued. By Lemma 2.2 and our hypothesis that $\tau(M')$ is deterministic, we know that M' must be single-valued, giving us the desired contradiction.

We show M' is not single-valued as follows: First note that, since M is refuted, M can go from any state to a final state in at most $(n-1)$ moves. Thus, from (*) above

$$q_0 \xrightarrow[M]{*} (w/x)p \xrightarrow[M]{*} (u/x_2)p_f$$

(**) $$q_0 \xrightarrow[M]{*} (w/y)q \xrightarrow[M]{*} (v/y_2)q_f$$

where p_f, $q_f \in F$, $|x_2| \leq (n-1)d$ and $|v| \leq n-1$.

Consider the action of M' on input $wv\$$

(***) $$q'_0 \xrightarrow[M']{*} (w/z_1)q' \xrightarrow[M']{*} (v\$/z_2)q'_f$$

where $q'_f \in F'$, z_1 is a prefix of xx_2, and $z_1z_2 = yy_2\$$. We compute the length of z_2 by

$$|z_2| = |z_1z_2| - |z_1| > |y| - |xx_2| > k - (n-1)d$$

and similarly

$$|v\$| = |v| + 1 \leq (n-1) + 1 = n.$$

Since the values of n and d are fixed (independent of f), the sequence

$$q' \xrightarrow[M']{*} (v\$/z_2)q'_f$$

from (***) satisfies the hypotheses of Lemma 3.1 for sufficiently large choice of k. Thus, M' cannot be deterministic, and we have the contradiction we sought. $\square$

Lemma 3.3: If $\tau(M)$ is deterministic then M has property 3.1(b).

Proof: Similar. $\square$

To show that properties 3.1(a) and (b) imply that $\tau(M)$ is deterministic, we construct a transducer M', with a possibly infinite number of states, which is equivalent to M. We then show that (a) and (b) imply that M' is deterministic, and that only finitely many of the states of M' are accessible. Let

$$M' = (\mathscr{C}, \Sigma \cup \{\$\}, \Delta \cup \{\$\}, \delta', C_0, \{C_f\})$$

where

$$\mathscr{C} \subseteq \mathscr{P}(Q \times \Delta^*) \cup \{C_f\}$$

$$C_0 = \{(q,x) \mid q_0 \xrightarrow[M]{*} (e/x)q\}$$

δ' is defined by

- if $C \subseteq Q \times d\Delta^*$ for some $d \in \Delta$, then

    1) $\delta'(C,e) = \{(C_1,d)\}$

        where $C_1 = \{(q,x) \mid (q,dx) \in C\}$.

- otherwise,

    2) $\delta'(C,a) = \{(C_2,e)\}$

        where $C_2 = (q,xyz) \mid \exists (p,x) \in C,\ r \in Q$ s.t.

$$p \xrightarrow[M]{} (a/y)r \xrightarrow[M]{*} (e/z)q$$

    3) $\delta'(C,\$) = \{(C_f,x\$) \mid (q_f,x) \in C,\ q_f \in F\}$.

We assume all inaccessible states are removed from $\mathscr{C}$.

<u>Lemma 3.4</u>: Whether or not M is single-valued and has Properties 3.1(a) and (b),

(i) if $C_0 \xrightarrow[M']{*} (w/x)C$ and $(q,y) \in C$

then $q_0 \xrightarrow[M]{*}$ [illegible]

(ii) if $q_0 \xrightarrow[M]{*} (w/z)q$

then $\exists x,y,C$ s.t. $C_0 \xrightarrow[M']{*} (w/x)C$, $(q,y) \in C \wedge z = xy$.

<u>Proof</u>: Both (i) and (ii) are proved by induction on $|w|$. The proofs are routine, and we omit them. □

<u>Lemma 3.5</u>: $\tau(M') = \{(w\$,x\$) \mid (w,x) \in \tau(M)\}$.

<u>Proof</u>: Straightforward from Lemma 3.4 above and the definition of type 3 moves of M', which take M' to its final state. □

<u>Lemma 3.6</u>: If M is single-valued then M' is deterministic.

<u>Proof</u>: M' is trivially deterministic for rules of type(1) and (2) in the definition of M'. For rules of type (3), suppose there exists C s.t. $|\delta'(C,\$)| > 1$; i.e.,

$$(C_f,y\$),\ (C_f,z\$) \in \delta'(C,\$), \quad y \neq z.$$

Choose w such that

$$C_0 \xrightarrow[M']{*} (w/x)C \xrightarrow[M']{*} (\$/y\$)C_f \text{ and } C_0 \xrightarrow[M']{*} (w/x)C \xrightarrow[M']{*} (\$,z\$)C_f$$

whence both $(w\$,xy\$)$ and $(w\$,xz\$)$ are in $\tau(M')$. But by Lemma 3.5 this implies M is not single-valued, a contradiction. □

<u>Lemma 3.6</u>: If M is single-valued and has Properties 3.1(a) and (b), then M' is a finite transducer.

<u>Proof</u>: Let k and ℓ be the bounds defined for properties 3.1(a) and (b). Let n = # states of $M = \#Q$, and d = max # output symbols emitted in any single move of M. Finally, let $m = \max(k,\ell) + nd$. We show by induction on i that if $C_0 \xrightarrow[M']{i} C$, then $C \in \mathcal{P}(Q \times \Delta^{*m}) \cup \{C_f\}$; thus the set of reachable states of M' is finite.

<u>Basis</u>: For $i=0$, $C = C_0 = \{(q,x) \mid q_0 \xrightarrow[M]{*} (\bar{e}/x)q\}$.
Consider any $(q,x) \in C_0$. Since M is single-valued, M has no cycles on e-input with non-empty output. Thus, x is the output of at most $n-1$ steps of M, and $|x| \leq (n-1)d$.

<u>Inductive Step</u>. Let $C_0 \xrightarrow[M']{*} C' \xrightarrow[M']{} C$. There are 3 cases, depending on whether the final move comes from rule (1), (2), or (3) in the definition of M'.

<u>rule 1</u>: $C' \xrightarrow[M']{} (e/a)C$. In this case $(q,x) \in C$ iff $(q,ax) \in C'$. By the inductive hypothesis, $C' \subseteq Q \times \Delta^{*m}$, thus $|ax| \leq m$, $|x| \leq m-1 < m$, and $C \subseteq Q \times \Delta^{*m}$.

<u>rule 2</u>: $C' \xrightarrow[M']{} (a/e)C$. We claim that if rule (2) is applicable to state C', then $C' \subseteq Q \times \Delta^{*\max(k,\ell)}$. If rule (2) is applicable $C' \not\subseteq Q \times a\Delta^*$ for any a. Thus, one of the following three cases applies:

(i) $C' \subseteq Q \times \{e\} \subseteq Q \times \Delta^{*\max(k,\ell)}$.

(ii) $\exists (q,e) \in C'$. Consider any $(p,y) \in C'$, and let $C_0 \xrightarrow[M]{*} (w/x)C'$. Then by Lemma 3.4, $q_0 \xrightarrow[M]{*} (w/x)q$

and $q_0 \xrightarrow[M]{*} (w/xy)p$, so by property 3.1 a ,
$|y| \leq k$.

(iii) for any $(p,ay) \in C'$, $\exists (q,bz) \in C'$, $b \neq a$. Again $q_0 \xrightarrow[M]{*} (w/xay)p$ and $q_0 \xrightarrow[M]{*} (w/xbz)q$. By property 3.1(b), $|ay| \leq \ell$.

Now, letting $C' \xrightarrow[M']{} (a/e)C$ by rule (2), we have $(q,y) \in C$ iff $(p,y_1) \in C'$, $p \xrightarrow[M]{} (a/y_2)p' \xrightarrow[M]{*} (e/y_3)q$ and $y = y_1y_2y_3$. By the above argument $|y_1| \leq \max(k,\ell)$; $|y_2| \leq d$ by definition; $|y_3| \leq (n-1)d$ since M has no nontrivial cycles. Thus $|y_1y_2y_3| \leq \max(k,\ell) + nd = m$, as desired.

rule 3: If $C' \xrightarrow[M']{} C$ by rule 3 , then $C = C_f$, and the lemma is satisfied trivially.

This completes the induction, proving that M' has only finitely many states. To complete the proof we need only show that δ' is finite - i.e., for any C and any $a \in \Sigma \cup \{e,\$\}$, $\delta'(C,a)$ is finite. Since M is single-valued†, this follows trivially from Lemma 3.5, which says M' is single-valued. □

---

† δ' is finite even if M is not single-valued: transitions due to rules (1) and (2) are trivially finite, and rule (3) obeys $|\delta'(C,\$)| \leq |C|$. Since we have shown that each state is a finite set (in addition to there being only finitely many states), the result follows.

Theorem 3.2: Let M be a (nd)ft. Then $\tau(M)$ is deterministic iff M is single-valued and properties 3.1(a) and (b) hold.

Proof: Immediate from Lemmas 3.2 thru 3.6. □

Now, having characterized the deterministic translations by these properties (single-valued, and 3.1(a) and (b)) we show how to decide these properties. A decision algorithm for single-valuedness was given in Theorem 2.2; algorithms to decide properties 3.1 appear below.

Lemma 3.7: A ndft $M = (Q,\Sigma,\Delta,\delta,q_0,f)$ violates property 3.1(a) iff $\exists u,v \in \Sigma^*$, $|uv| \leq |Q|^2$, $q_0 \xrightarrow[M]{*} (u/w)p \xrightarrow[M]{*} (v/x)p$, $q_0 \xrightarrow[M]{*} (u/y)q \xrightarrow[M]{*} (v/z)q$ and $|x| > |z|$.

Proof: If: clearly $q_0 \xrightarrow[M]{*} (uv^n/wx^n)p$ and $q_0 \xrightarrow[M]{*} (uv^n/yz^n)q$, and $|wx^n| - |yz^n|$ grows unboundedly with n.

only if: We show by induction on # moves of M in a shortest violation of Property 3.1(a).

Basis: Fewer than $|Q|^2$ moves, trivial.

Induction: Suppose $q_0 \xrightarrow[M]{*} (u/w)p$, $q_0 \xrightarrow[M]{*} (u/y)q$ and $|w| - |y| > k$. In the first $|Q|^2$ moves, some pair of states must repeat - i.e.,

r,s $\in$ Q s.t.

$$q_0 \xrightarrow[M]{*} (u_1/w_1)r \xrightarrow[M]{*} (u_2/w_2)r \xrightarrow[M]{*} (u_3/w_3)p$$

$$\underbrace{q_0 \xrightarrow[M]{*} (u_1/y_1)}_{A}\underbrace{s \xrightarrow[M]{*} (u_2/y_2)}_{B}\underbrace{s \xrightarrow[M]{*} (u_3/y_3)}_{C}q$$

where $u = u_1u_2u_3$, $w = w_1w_2w_3$, $y = y_1y_2y_3$, and $|u_1u_2| \leq |Q|^2$. Either

$|w_2| \neq |y_2|$ - in which case the lemma follows immediately from the computation Ⓐ Ⓑ - or else $|w_2| = |y_2|$ so the computation Ⓐ Ⓒ also has outputs whose lengths differ by k; Ⓐ Ⓒ is shorter than Ⓐ Ⓑ Ⓒ, so by the inductive hypothesis the lemma follows. □

<u>Lemma 3.3</u>: It is decidable whether a ndft satisfying Property 3.1(a) also satisfies Property 3.1(b).

<u>Proof</u>: The following algorithm suffices.

1) compute the collection of all triples (p,q,u) s.t. for some $w \in \Sigma^*$, $q_0 \xrightarrow[M]{*} (w/x)p$ and $q_0 \xrightarrow[M]{*} (w/xu)q$. Property 3.1(a) guarantees that this set is finite; computing it is straightforward.

2) for each $q \in Q$ and $a \in \Delta$, construct the sets

   $Lq,a = \{w \mid q \xrightarrow[M]{*} (w/x)p$ for some $p \in Q$, $x \in \Delta^*$ s.t. $1{:}x = a\}$

   $Lq,e = \{w \mid q \xrightarrow[M]{*} (w/e)p$ for some $p \in Q\}$

   $$Lq = \bigcup_{a \in \Delta \cup \{e\}} Lq,a$$

   These sets are effectively regular.

3) for each $q \in Q$ and $a \in \Delta$, we define

   $Mq,a(w) = \{x \mid q \xrightarrow[M]{*} (w/x)p$ for some $p \in Q$, $x \in \Delta^*$ s.t. $1{:}x = a\}$

   and for any $L \subseteq \Sigma^*$,

   $$Mq,a(L) = \bigcup_{w \in L} Mq,a(w)$$

   For any regular L, Mq,a(L) is effectively regular.

4) Now, M satisfies Property 3.1(b) iff

(i) for each triple (p,q,u) constructed in (1), with $u \neq e$, for each $a \in \Delta$, $a \neq 1{:}u$, $M_{p,a}[L_{p,a} \cap L_q] \cup M_{q,a}[L_{p,a} \cap L_q]$ is finite.

(ii) for each triple (p,q,e) and each $a,b \in \Delta$, $a \neq b$, $M_{p,a}[L_{p,a} \cap L_{q,b}] \cup M_{q,b}[L_{p,a} \cap L_{q,b}]$ is finite. (Note if (p,q,e) is reachable, then so are (p,p,e), (q,q,e), and (q,p,e).) □

<u>Theorem 3.2</u>: It is decidable whether a ndft encipherer M has a deterministic finite decipherer; moreover, if such a decipherer exists we can effectively find it.

<u>Proof</u>: Immediate from above lemmas. □

4. <u>Pushdown Transducer Encipherers</u>

While enciphering devices that cannot be realized by finite transducers are probably not of practical importance, it is of some theoretical interest to note that for slightly more powerful devices even the basic question of ambiguity is undecidable. Proof of an equivalent result appears in a very different context in [Br, Theorem 5.2.5]. We use the following standard definitions [A & U].

<u>Definition 4.1</u>: A pushdown transducer (PDT), P, is an 8-tuple $(Q,\Sigma,\Gamma,\Delta,\delta,q_0,Z_0,F)$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite alphabet of pushdown list

symbols, Δ is a finite output alphabet, δ is a mapping from $Q \times (\Sigma \cup \{e\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^* \times \Delta^*$, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is the symbol that appears initially on the pushdown list, and $F \subseteq Q$ is the set of final states.

A configuration of P is a 4-tuple (q,x,α,y), where q represents the current state, x represents the unused portion of the input, α represents the contents of the pushdown list, and y is the output string emitted to this point. If $\delta(q,a,Z)$ contains (r,α,z), then we write $(q,ax,Z\gamma,y) \vdash (r,x,\alpha\gamma,yz)$ for all $x \in \Sigma^*$, $\gamma \in \Gamma^*$, and $y \in \Delta^*$. The translation defined by P, denoted $\tau(P)$, is $\{(x,y) \mid (q_0,x,Z_0,e) \overset{*}{\vdash} (q,e,\alpha,y)$ for some $q \in F$ and $\alpha \in \Gamma^*\}$.

We say that P is deterministic (a DPDT) when (1) for all $q \in Q$, $a \in \Sigma \cup \{e\}$, and $Z \in \Gamma$, $\delta(q,a,Z)$ contains at most one element, and (2) if $\delta(q,e,Z) \neq \phi$, then $\delta(q,a,Z) = \phi$ for all $a \in \Sigma$.

<u>Theorem 4.1</u>: For ciphers generated by DPDT encipherers ambiguity is undecidable.

Proof: Let $A = x_1, x_2, \ldots, x_n$; $B = y_1, y_2, \ldots, y_n$ be an instance of Post's Correspondence Problem. (See [H & U]).

Consider the translation

$$T = \{(\$\, i_1 i_2 \ldots i_k,\ x_{i_1} x_{i_2} \ldots x_{i_k} \# i_k i_{k-1} \ldots i_1) \mid i_j \in \{1,2,\ldots m\}\}$$
$$\cup\ \{(¢\, i_1 i_2 \ldots i_k,\ y_{i_1} y_{i_2} \ldots y_{i_k} \# i_k i_{k-1} \ldots i_1) \mid i_j \in \{1,2,\ldots m\}\}.$$

It is not difficult to see that there is a DPDT, P, such that

$\tau(P) = T$. Furthermore, the cipher generated when P is viewed as an enciphering device is ambiguous if and only if there exist $i_1, i_2, \ldots, i_k \in \{1,2,\ldots,m\}$ such that $x_{i_1}x_{i_2}\ldots x_{i_k}\# i_k i_{k-1}\ldots i_1 = y_{i_1}y_{i_2}\ldots y_{i_1}\# i_k i_{k-1}\ldots i_1$. Thus the cipher is ambiguous if and only if this instance of Post's Correspondence Problem has a solution. Hence, ambiguity is undecidable for DPDT encipherers.

References

1. [A&U] A.V. Aho and J.D. Ullman, The Theory of Parsing, Translation, and Compiling, Vols. 1 and 2, Prentice-Hall, 1973.

2. [Bird] Malcom Bird, "The Equivalence Problem for Deterministic Two-Tape Automata", JCSS, 7:2, April 19[illegible].

3. [B] Ingrid Bruckner, "Fur Konstruktion von Decodierautomatem".

4. [H] Lance Hoffman, Modern Methods for Computer Privacy and Security, Prentice-Hall, to appear.

5. [H1] J.E. Hopcroft, "On The Equivalence and Containment Problems for Context-Free Languages", Mathematical Systems Theory, 3:2.

6. [H2] J.E. Hopcroft, private communication.

7. [H&U] J.E. Hopcroft and J.D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.

8. [K] D. Kahn, The Codebreakers, Macmillan Co., 1967.

9. [NBS] "Guidelines for implementing and using the NBS data encryption standard", National Bureau of Standards, 1975.

10. [S&P] A.A. Sardinas and G.W. Patterson, "A necessary and sufficient condition for unique decomposition of encoded messages", IRE Convention Record, pt. 8, pp.104-108, 1953.

11. [T] R. Turn, "Privacy transformations for data bank systems", Proc. National Computer Conference, 1973, pp.589-601.