

# Sigaba

# Sigaba

- ❑ Used by Americans during WWII
  - And afterwards (to about 1948)
- ❑ Never broken
  - Germans quit collecting, considered impossible
- ❑ Like Enigma, Sigaba is a rotor machine
  - But instead of 3 rotors, Sigaba uses 15 rotors!
- ❑ Not suitable for battlefield
  - Bigger, heavier, more fragile than Enigma

# Sigaba

- ❑ Like a big typewriter
- ❑ A really big typewriter...
- ❑ A really, really big typewriter...

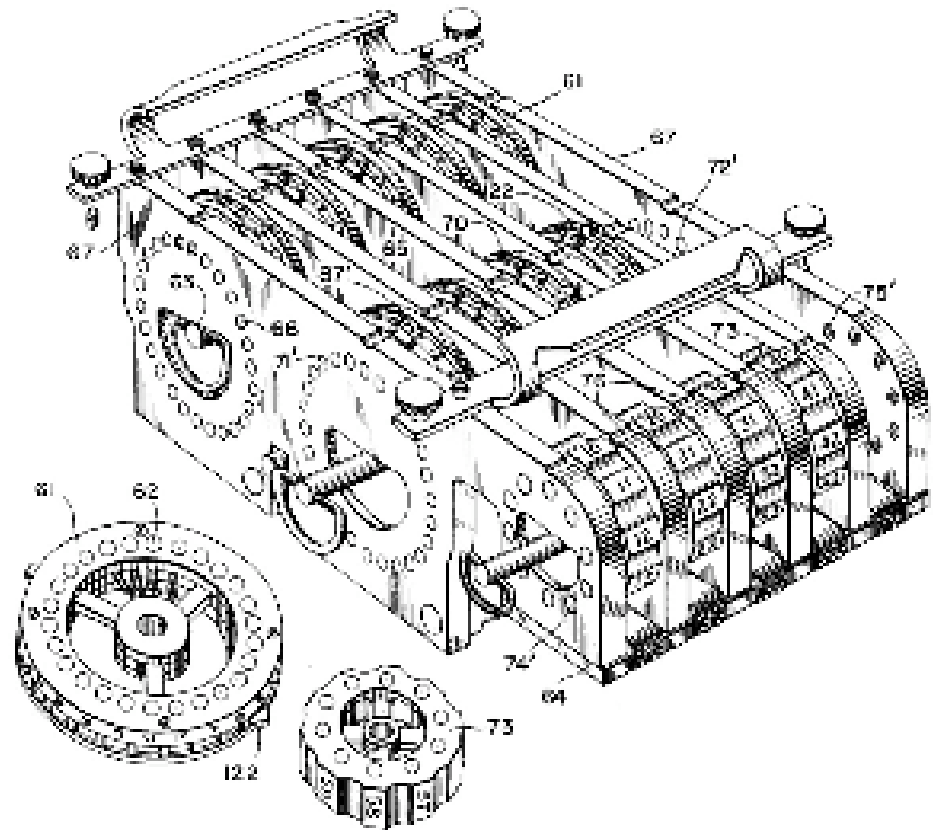


# Sigaba

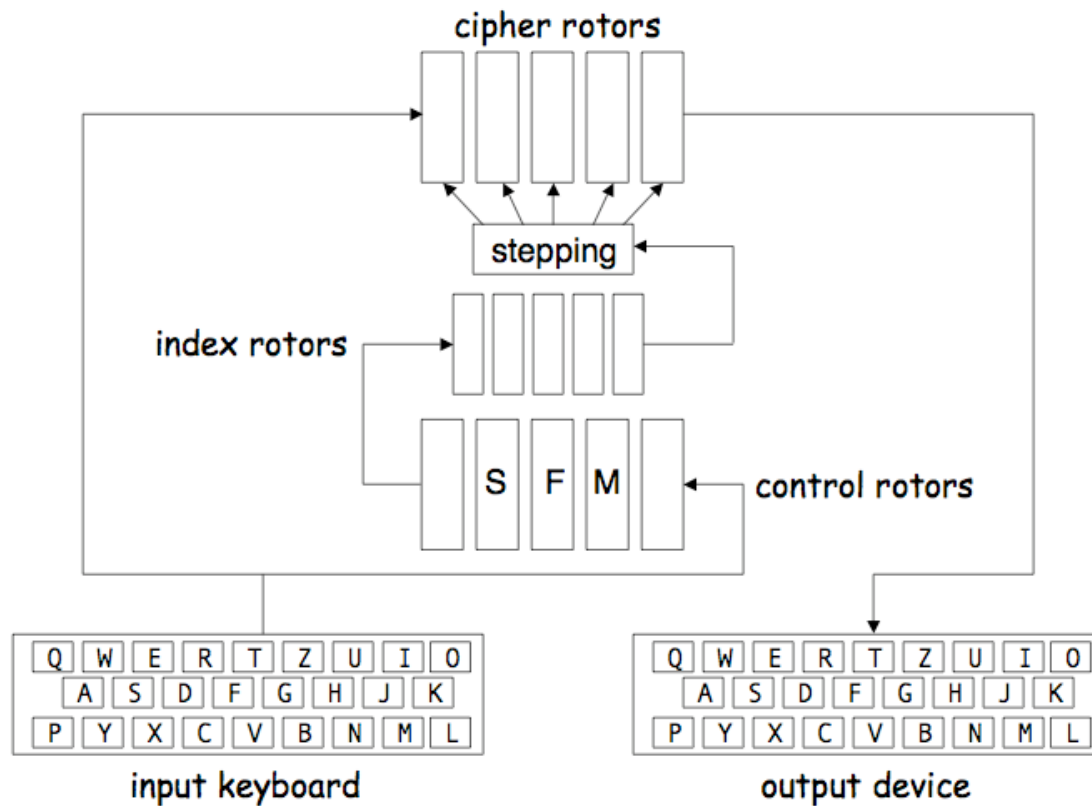
- ❑ Developed by Rowlett, Friedman and others
  - Knew odometer stepping of rotors is weakness
- ❑ Sigaba uses 10 rotors to determine stepping of the 5 cipher rotors
- ❑ Any of 5 cipher rotors can step
  - From 1 to 4 cipher rotors step for each letter
- ❑ Almost like using one Enigma to determine rotor stepping of another Enigma

# Sigaba

- ❑ Rotors
- ❑ Lots of rotors...
- ❑ Rotors, rotors, and more rotors

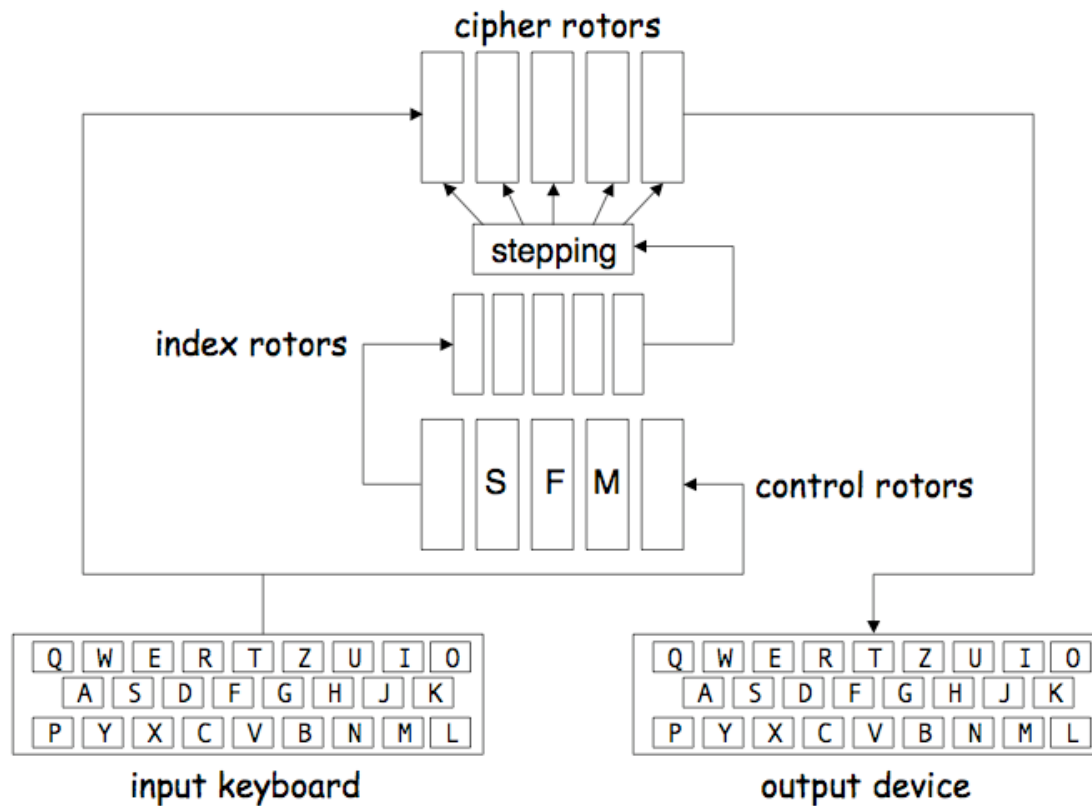


# Sigaba Wiring Diagram



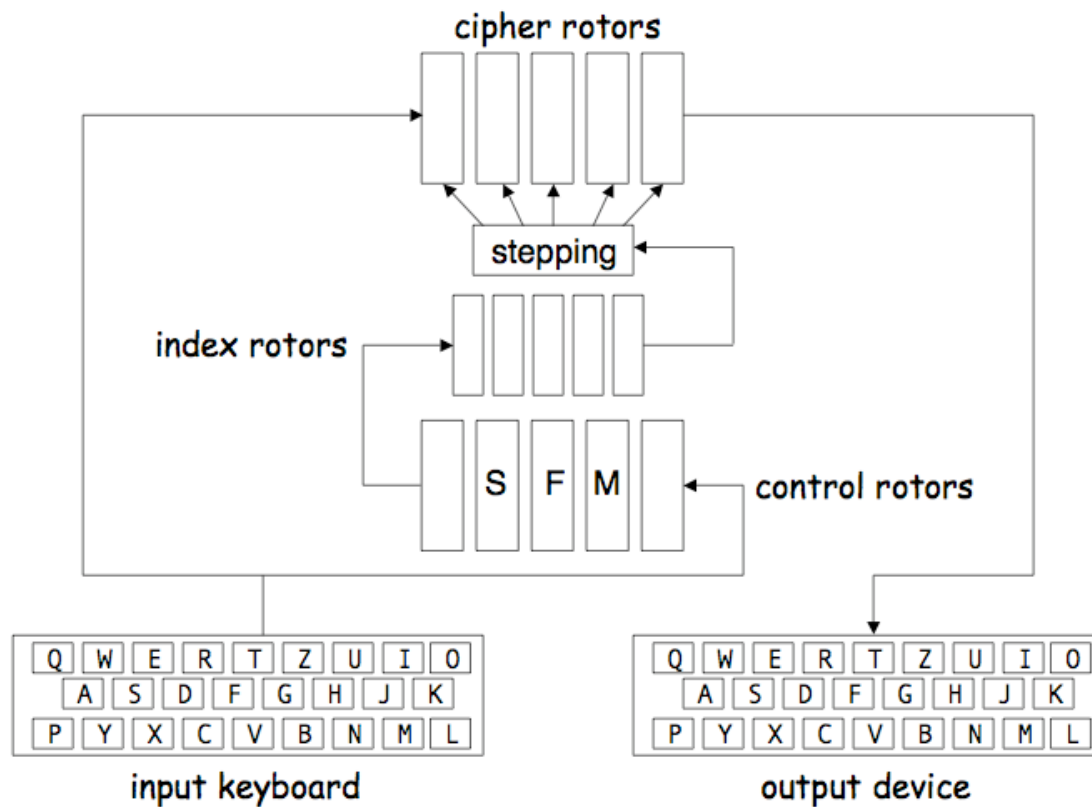
- Control and index rotors determine stepping of cipher rotors

# Sigaba Wiring Diagram



- ❑ Control and cipher rotors each permute 26 letters
- ❑ Can be interchanged and inserted in reverse

# Sigaba Wiring Diagram



- ❑ Control rotors step like "scrambled odometer"
- ❑ Index rotors set initially, but do **not** step
- ❑ From 1 to 4 of cipher rotors step



# Rotor Stepping

- ❑ Index rotors do not step
- ❑ Control rotors
  - Middle 3 step as: slow, fast, medium
  - Outside rotors don't step
- ❑ Cipher rotors
  - At least 1, at most 4 step each time
  - Stepping is somewhat complex...

# Cipher Rotor Stepping

- ❑ When a letter is typed
- ❑ 4 inputs to **control rotors** activated
  - These are: F , G , H , I
  - Then 4 (scrambled) letters output
- ❑ Outputs of control rotors combined
  - Then fed into index rotors

# Cipher Rotor Stepping

- ❑ Outputs of control rotors combined
  - Result(s) go into index rotors
  - From 1 to 4 inputs to index rotors
- ❑ Index rotor outputs combined in pairs
  - Active index rotor outputs determine which cipher rotors step

# Cipher Rotor Stepping

- F , G , H , I input to control rotors
- Let  $I_0, I_1, \dots, I_9$  be inputs to index rotors
  - $I_0$  is always inactive, and...

$I_1 = B$	$I_2 = C$	$I_3 = D \vee E$
$I_4 = F \vee G \vee H$	$I_5 = I \vee J \vee K$	$I_6 = L \vee M \vee N \vee O$
$I_7 = P \vee Q \vee R \vee S \vee T$	$I_8 = U \vee V \vee W \vee X \vee Y \vee Z$	$I_9 = A$

- Where “ $\vee$ ” is OR, and
  - A,B,C,..., Z are control rotor **outputs**

# Cipher Rotor Stepping

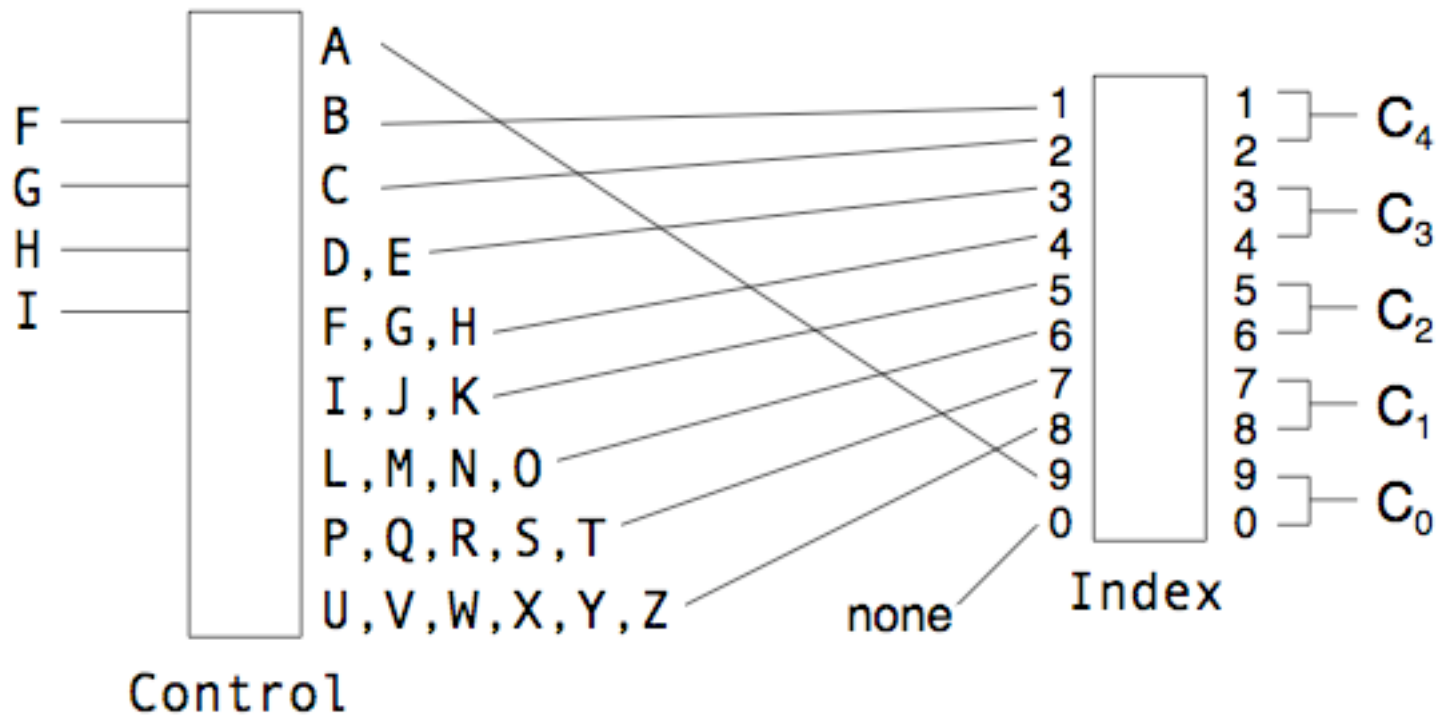
- Let  $O_0, O_1, \dots, O_9$  be index rotor outputs
- If  $C_i == 1$ , cipher rotor  $i$  steps
  - Cipher rotors numbered left-to-right
- Then the  $C_i$  given by

$C_0 = O_0 \vee O_9$	$C_1 = O_7 \vee O_8$	$C_2 = O_5 \vee O_6$	$C_3 = O_3 \vee O_4$	$C_4 = O_1 \vee O_2$
----------------------	----------------------	----------------------	----------------------	----------------------

- Note that 1 to 4 of the  $O_i$  are active
- Implies that 1 to 4 of  $C_i$  are active

# Stepping Maze

- A picture is worth  $2^{10}$  words ?



# Theoretical Keyspace

- ❑ If cipher/control rotors all set to "A"
- ❑ And index rotors all set to "0"
  - Select 5 cipher rotors:  $(26!)^5 = 2^{442}$
  - Select 5 control rotors:  $(26!)^5 = 2^{442}$
  - Select 5 index rotors:  $(10!)^5 = 2^{109}$
- ❑ Keyspace is enormous: 993 bits!

# WWII Sigaba Keyspace

- ❑ 10 cipher and control rotors
  - 2 orientations for each
  - Control rotors set to any initial position
  - Cipher rotors set to default positions (usually)
- ❑ 5 index rotor
  - Inserted in a fixed order
  - Each set to one of 10 initial positions
- ❑ Apparently, gives  $10! \cdot 2^{10} \cdot 26^5 \cdot 10^5 = 2^{71.9}$



# WWII Actual Keyspace

- ❑ Keyspace of  $10! \cdot 2^{10} \cdot 26^5 \cdot 10^5 = 2^{71.9}$  ?
- ❑ Control rotors settings sent in the clear!
  - Sent as part of the message indicator (MI)
  - An attacker could see this
- ❑ So, actual keyspace:  $10! \cdot 2^{10} \cdot 10^5 = 2^{48.6}$ 
  - On POTUS-PRIME link, 71.9-bit keyspace used
- ❑ Why give away so much of keyspace?

# WWII Actual Keyspace

- ❑ Why give away so much of keyspace?
- ❑ Device is complex to configure
- ❑ Larger keyspace means
  - More settings to initialize
  - More chance for error
  - Problem for time-sensitive info
- ❑ One daily key
  - Then MI gives unique message key

# WWII Theoretical Keyspace

- ❑ Given components available in WWII
- ❑ Appears that maximum keyspace was  $10! \cdot 2^{10} \cdot 26^{10} \cdot 5! \cdot 10^5 = 2^{102.3}$
- ❑ But this is a little too high
  - Index rotors do not rotate
  - So only  $10!$  different index perms
  - And  $10! < 5! \cdot 10^5$
- ❑ So it looks like largest possible keyspace  $10! \cdot 2^{10} \cdot 26^{10} \cdot 10! = 2^{100.6}$

# WWII Theoretical Keyspace

- ❑ About  $10! \cdot 2^{10} \cdot 26^{10} \cdot 10! = 2^{100.6}$  ?
- ❑ No, this is still too high!
- ❑ Index rotor outputs are ORed in pairs
  - Used to determine cipher rotor stepping
  - As seen previously
- ❑ Therefore, 32 equivalent index perms
- ❑ Final answer: maximum WWII keyspace  
 $10! \cdot 2^{10} \cdot 26^{10} \cdot 10!/32 = 2^{95.6}$

# Sigaba Attack

## □ Assumptions

- Full 95.6 bit WWII keyspace is used
- Trudy has a Sigaba machine (so Trudy knows rotor permutations)
- Trudy has some known plaintext
- Goal: use as little known plaintext as possible

## □ How efficiently can we attack Sigaba under these assumptions?

# Sigaba Attack

- ❑ Two phases to this (complex) attack
- ❑ **Primary phase**
  - Find all cipher rotor settings that are consistent with known plaintext
  - Requires some amount of known plaintext
- ❑ **Secondary phase**
  - Find control rotor settings, index perm
  - May require more known plaintext

# Primary Phase

- ❑ Select and initialize cipher rotors
  - $\text{Binomial}(10,5) \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{43.4}$  settings
- ❑ For each of these  $2^{43.4}$  settings, how many cipher perms are possible at next step?
  - From 1 to 4 cipher rotors can step
  - 30 ways that 1 to 4 (out of 5) rotors can step
- ❑ How can we use this information?

# Primary Phase

- ❑ Given a putative cipher rotor setting...
- ❑ Check whether it matches known plaintext
- ❑ If not, discard it
- ❑ If a match, try all 30 steps and keep any that match with next known plaintext
- ❑ Repeat until either
  - No matches (putative setting is discarded)
  - Used all known plaintext (save for secondary)



# Primary Phase

- ❑ Correct rotor setting is said to be **causal**
  - All incorrect settings are **random**
- ❑ How many random matches do we expect?
- ❑ First letter matches with probability  $1/26$
- ❑ If it matches, then must try all 30 steps
  - Each matches with probability  $1/26$
  - Binomial distribution with  $p = 1/26$  and  $n = 30$
- ❑ Expected matches is  $30/26 = 1.154$
- ❑ If first letter matches, then after  $n$  steps, we expect about  $(1.154)^n$  surviving paths

# Primary Phase

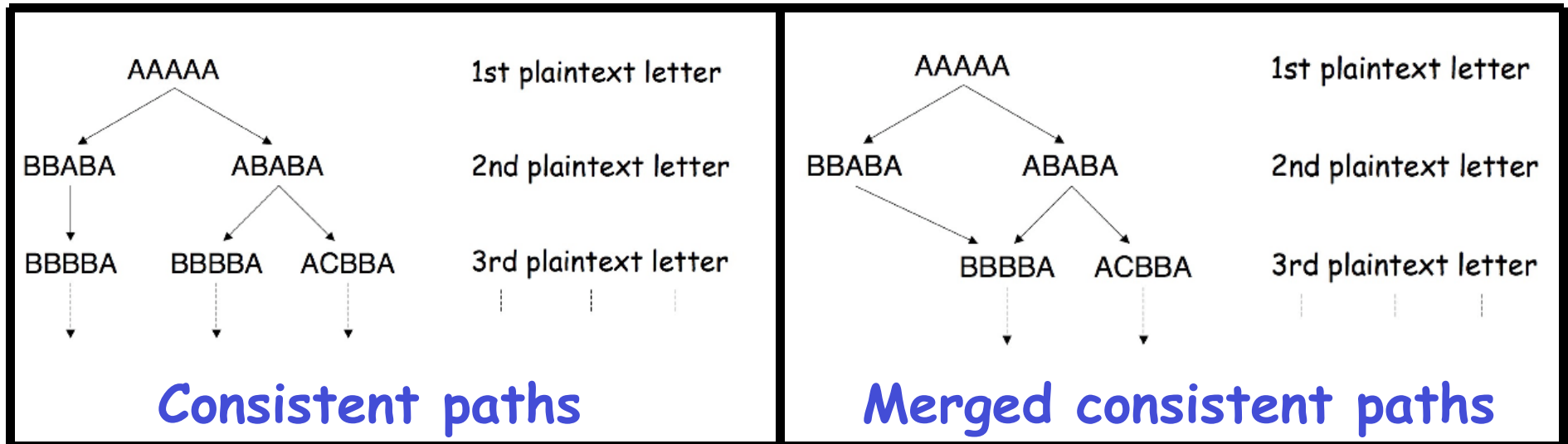
- ❑ If first plaintext matches, about  $(1.154)^n$  surviving paths after  $n$  steps
- ❑ This looks bad...
- ❑ We want to eliminate putative cipher rotor settings that are incorrect
  - The random settings
- ❑ How to do this when we get more paths!

# Primary Phase

- ❑ Maybe not as bad as it seems...
- ❑ We are only concerned with initial cipher rotor guess, not paths
- ❑ In fact
  - Some paths merge
  - Expected distribution of paths differs in causal case and random cases

# Primary Phase: Merging Paths

- Suppose first 3 plaintext match
  - With initial settings AAAAA



- Can merge paths since only the rotor settings (not path) needed for next step

# Primary Phase: Distributions

- ❑ In causal and random cases, expected number of paths differs
- ❑ Empirical results show that...

steps	average	maximum	tests	non-zero
10	6.5	27	100,000	763
20	11.8	56	100,000	516
30	16.5	84	100,000	427
40	20.8	105	100,000	324
50	28.4	194	100,000	290
60	38.8	163	100,000	275
70	47.1	415	100,000	269
80	71.3	524	100,000	212
90	77.6	486	100,000	216
100	100.5	1005	100,000	203

Random

steps	average	maximum	minimum	tests
10	10.2	51	1	10,000
20	19.6	94	1	10,000
30	29.6	151	1	10,000
40	40.1	237	1	10,000
50	54.1	404	1	10,000
60	69.2	566	1	10,000
70	85.0	689	1	5,000
80	105.0	829	2	5,000
90	130.4	1152	1	3,000
100	161.1	1926	1	3,000

Causal

- ❑ However, the variance is high

# Primary Phase: Bottom Line

- ❑ Test each of  $2^{43.4}$  cipher rotor settings
- ❑ Only 1/26 match 1st plaintext letter
- ❑ Many others eliminated due to merging
- ❑ More eliminated by expected distribution
  - Note: this makes the attack probabilistic
- ❑ Can reduce primary survivors to about  $2^{20}$

# Secondary Phase

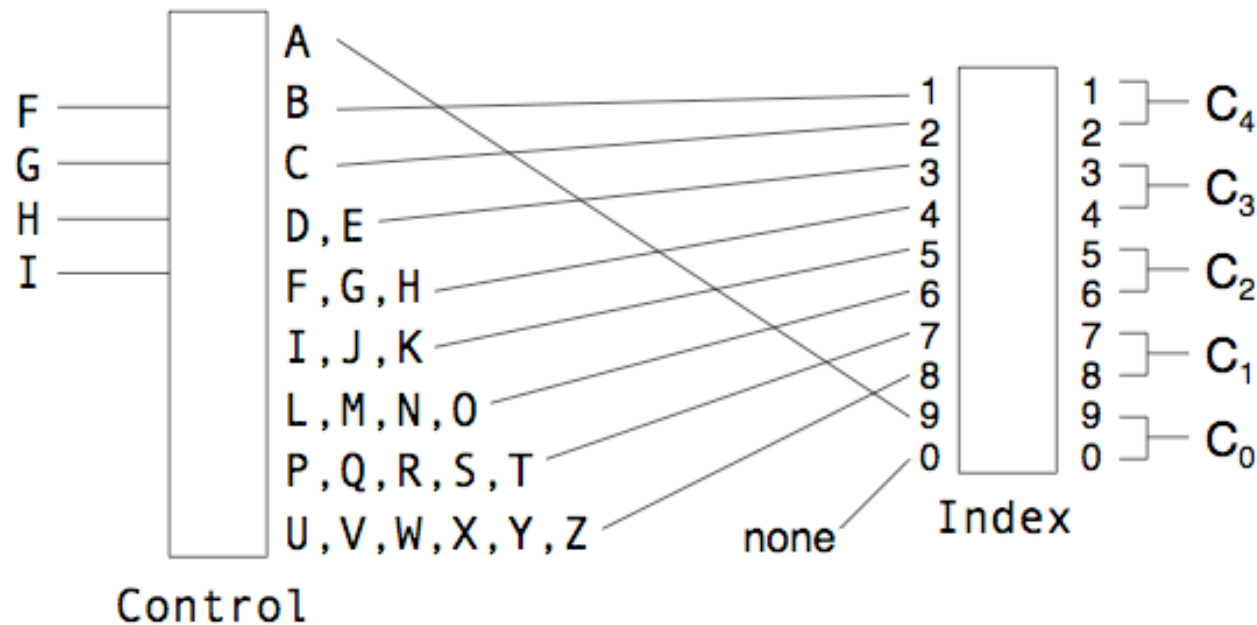
- ❑ Discussion here applies to each primary phase survivor
- ❑ Each primary survivor gives putative cipher rotors and settings
- ❑ Obvious secondary test is to...
  - Try all control and index settings:  
 $10!/32 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{52.2}$  of these
  - Work of more than  $2^{52}$  per primary survivor
- ❑ Can we do better?

# Secondary Phase

- ❑ Primary work is about  $2^{43}$ 
  - With about  $2^{20}$  survivors
- ❑ Obvious secondary has total work about  $2^{72}$ 
  - Since  $2^{52}$  work for each of  $2^{20}$  survivors
- ❑ Can we improve secondary phase?
- ❑ Cipher rotor motion is not uniform
  - Recall that from 1 to 4 steps for each letter
  - Also, index permutation is fixed for a message

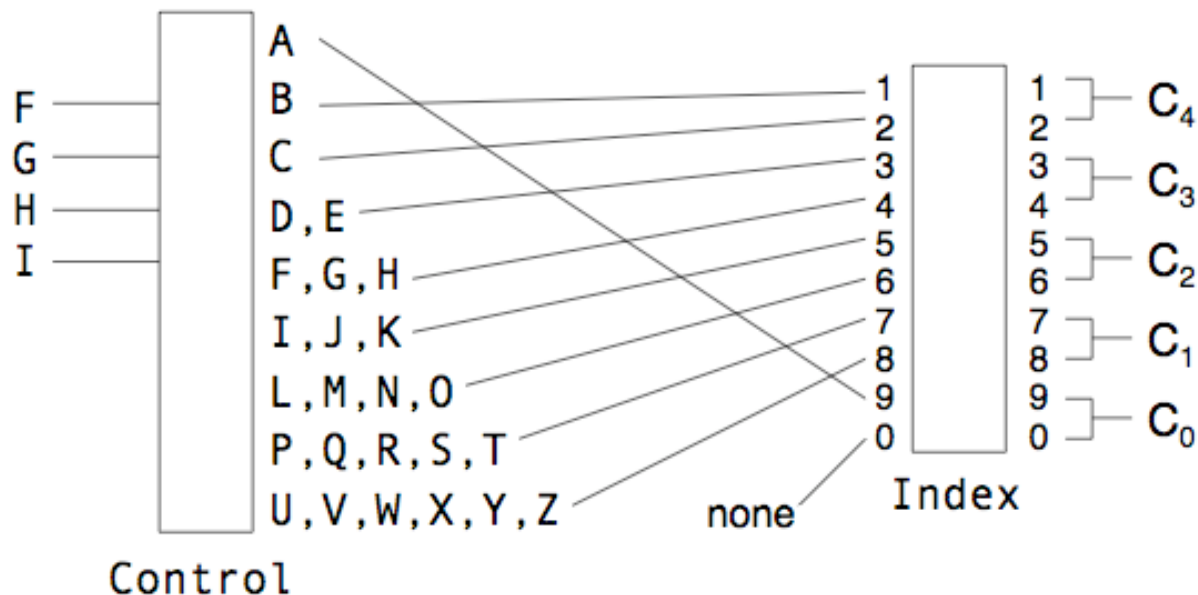


# Stepping Maze



- ❑ Model control permutations as random
- ❑ Probabilities of the  $C_i$  are not uniform

# Example



- ❑ Consider index perm 0123456789  $\rightarrow$  5479381026
  - $C_4$  connected to 10 control letters
  - $C_2$  connected to 1 control letter
- ❑  $C_4$  will step much more frequently than  $C_2$

# Index Perm Inputs

number of letters	count	pairs
1	3	(0,1) (0,2) (0,9)
2	4	(0,3) (1,2) (1,9) (2,9)
3	5	(0,4) (0,5) (1,3) (2,3) (3,9)
4	7	(0,6) (1,5) (2,5) (5,9) (1,4) (2,4) (4,9)
5	6	(0,7) (1,6) (2,6) (6,9) (3,4) (3,5)
6	6	(0,8) (1,7) (2,7) (7,9) (3,6) (4,5)
7	6	(1,8) (2,8) (8,9) (3,7) (4,6) (5,6)
8	3	(3,8) (4,7) (5,7)
9	3	(4,8) (5,8) (6,7)
10	1	(6,8)
11	1	(7,8)

- ❑ Can use this table to determine input pairs
  - Count number of times each cipher rotor steps (using the known plaintext)

# Improved Secondary

- ❑ About 100 to 200 known plaintexts
- ❑ Reduces number of index perms to...
  - ...about  $2^7$
- ❑ This reduces secondary work from
  - $10!/32 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{52.2}$
- ❑ To about
  - $2^7 \cdot 5! \cdot 2^5 \cdot 26^5 = 2^{42.4}$
- ❑ Note: this work is per primary survivor

# Sigaba Attack: Bottom Line

- ❑ WWII Sigaba had a readily available keyspace of 95.6 bits
- ❑ Our attack has work factor of about  $2^{60}$ 
  - Under reasonable assumptions
- ❑ Sigaba as generally used in WWII had exhaustive key search work of  $2^{47.6}$

# Bottom Bottom Line

- ❑ Given limitations of WWII, our attack shows Sigaba has, at most, 60 bits of security
- ❑ Although 95.6 bits of key available, only 47.6 generally used
- ❑ However, on link between Roosevelt and Churchill (POTUS-PRIME), 71.9 bit key used
- ❑ It would not make sense to have bigger key than work for reasonable shortcut attack

# Bottom Bottom Bottom Line

- ❑ Our attack is less work than exhaustive key search on POTUS-PRIME link
- ❑ Conclusion?
- ❑ Developers of Sigaba (Rowlett and Friedman) were unaware of the attack
- ❑ Or they did not consider it practical