

Random Numbers

Our goal is to turn a plaintext into a ciphertext that looks *random*

Random Numbers

Our goal is to turn a plaintext into a ciphertext that looks *random*
... but isn't really random

Random Numbers

Our goal is to turn a plaintext into a ciphertext that looks *random*
... but isn't really random

Doesn't that sound like a pseudo-random number generator (PRNG)?


PRNGs

```
> random(256)
243
> random(256)
122
> random(256)
187
> random(256)
43
> random(256)
200
> random(256)
229
```


PRNGs

```
> random.seed(74658319934)
> random(256)
243
> random(256)
122
> random(256)
187
> random(256)
43
> random(256)
200
> random(256)
229
```


Stream Cipher

Let $K = K_0, K_1, \dots, K_i, \dots$ be a sequence of random numbers that is generated from a seed 


Stream Cipher

Let $K = K_0, K_1, \dots, K_i, \dots$ be a sequence of random numbers that is generated from a seed 

The i th ciphertext CT element from the i th plaintext PT element:

$$CT_i = PT_i \oplus K_i$$



Stream Cipher

Let $K = K_0, K_1, \dots, K_i, \dots$ be a sequence of random numbers that is generated from a seed 

The i th ciphertext CT element from the i th plaintext PT element:

$$CT_i = PT_i \oplus K_i$$

Stream Cipher

Let $K = K_0, K_1, \dots, K_i, \dots$ be a sequence of random numbers that is generated from a seed 


The i th ciphertext CT_i element from the i th plaintext PT_i element:

$$CT_i = PT_i \oplus K_i$$

Since \oplus is its own inverse:

$$PT_i = CT_i \oplus K_i$$

Stream Cipher

Let $K = K_0, K_1, \dots, K_i, \dots$ be a sequence of random numbers that is generated from a seed 

The i th ciphertext CT element from the i th plaintext PT element:

$$CT_i = PT_i \oplus K_i$$

Since \oplus is its own inverse:

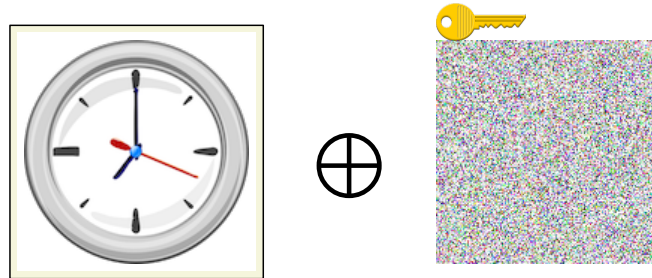
$$PT_i = CT_i \oplus K_i$$

Treating \oplus as mapping over sequences

$$CT = PT \oplus K$$

$$PT = CT \oplus K$$

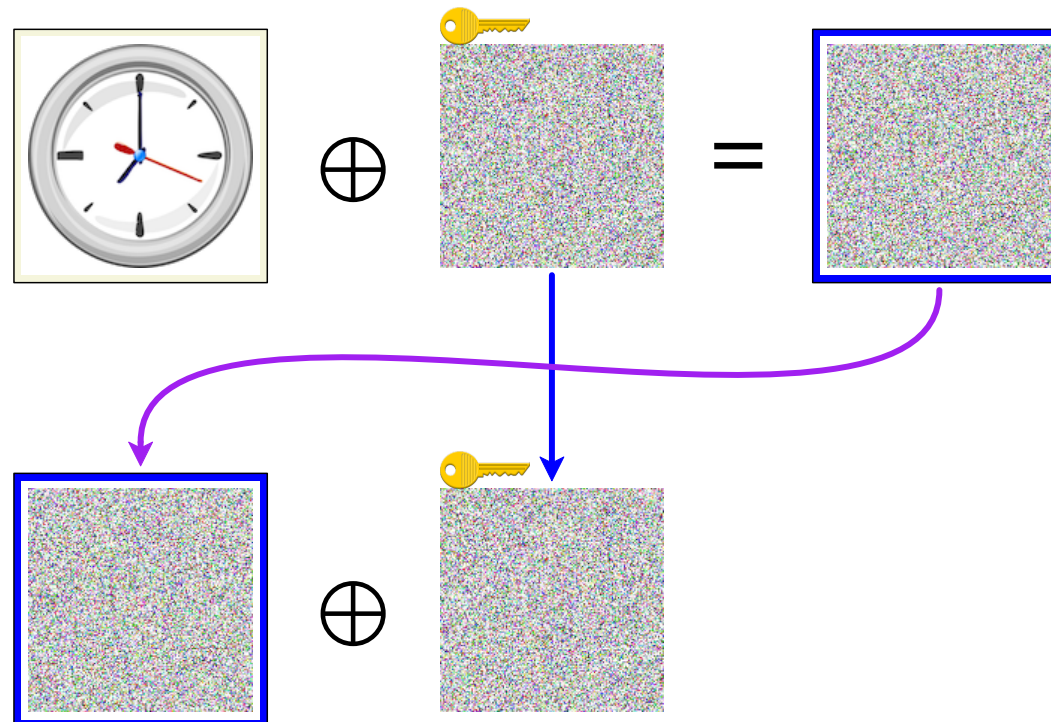
Stream Cipher on an Image



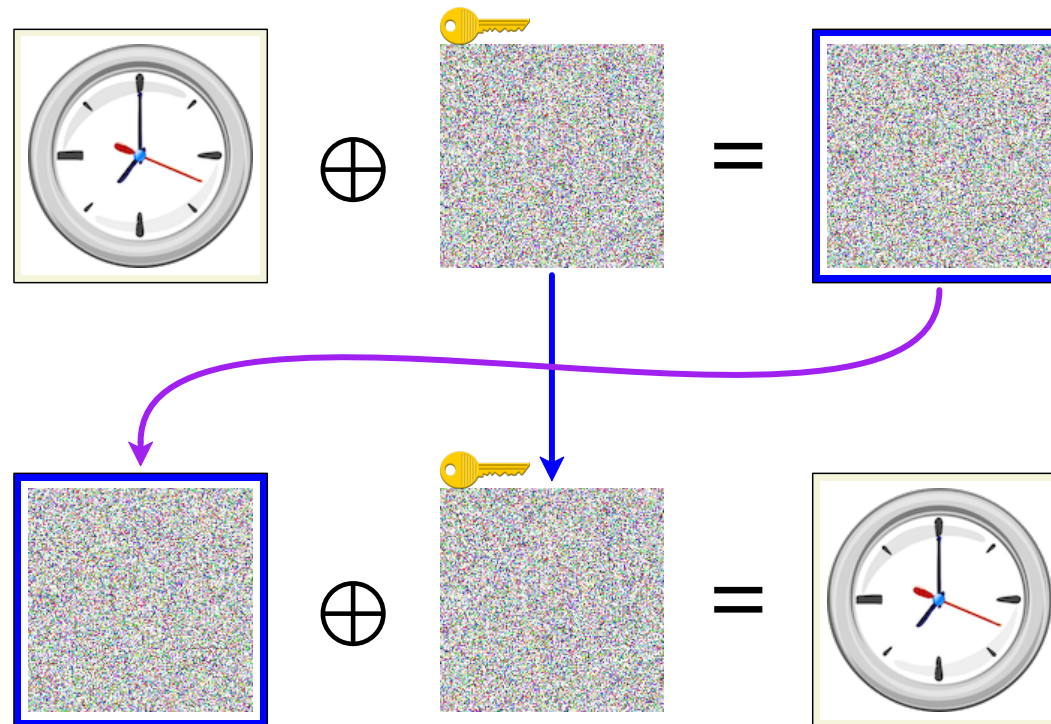
Stream Cipher on an Image



Stream Cipher on an Image



Stream Cipher on an Image



Well, That Was Easy!

Why aren't we done?

Well, That Was Easy!

Why aren't we done?

Problem #1: it turns out that \oplus uses up a key
unless the PRNG has an extra feature making it more general

Well, That Was Easy!

Why aren't we done?

Problem #1: it turns out that \oplus uses up a key
unless the PRNG has an extra feature making it more general

Problem #2: we're getting confidentiality, but not integrity
so, need to combine with something else, not today

Well, That Was Easy!

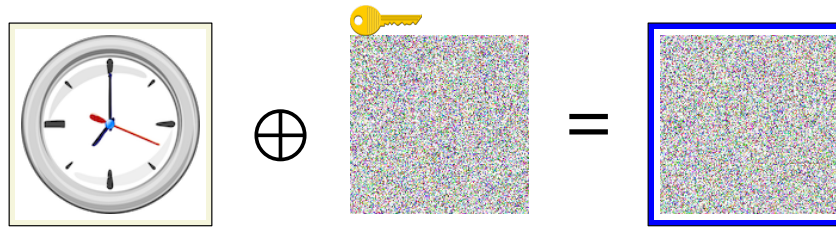
Why aren't we done?

Problem #1: it turns out that \oplus uses up a key
unless the PRNG has an extra feature making it more general

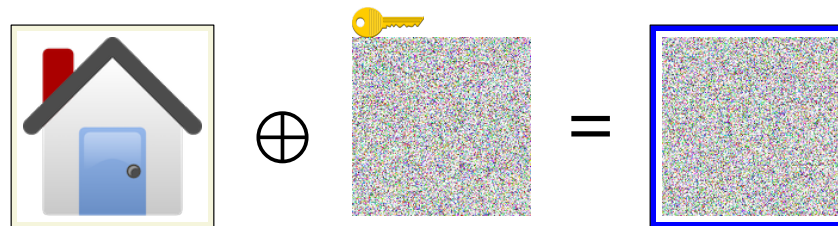
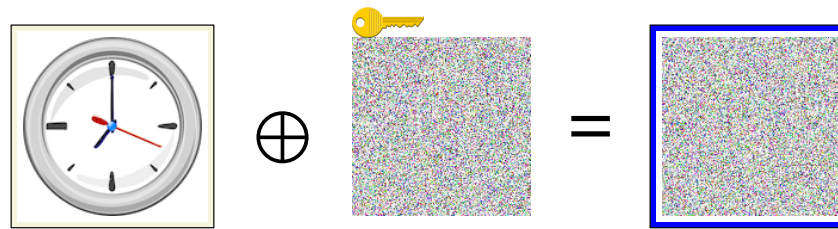
Problem #2: we're getting confidentiality, but not integrity
so, need to combine with something else, not today

Problem #3: we need a really good PRNG
a general building block, so we'll look more today

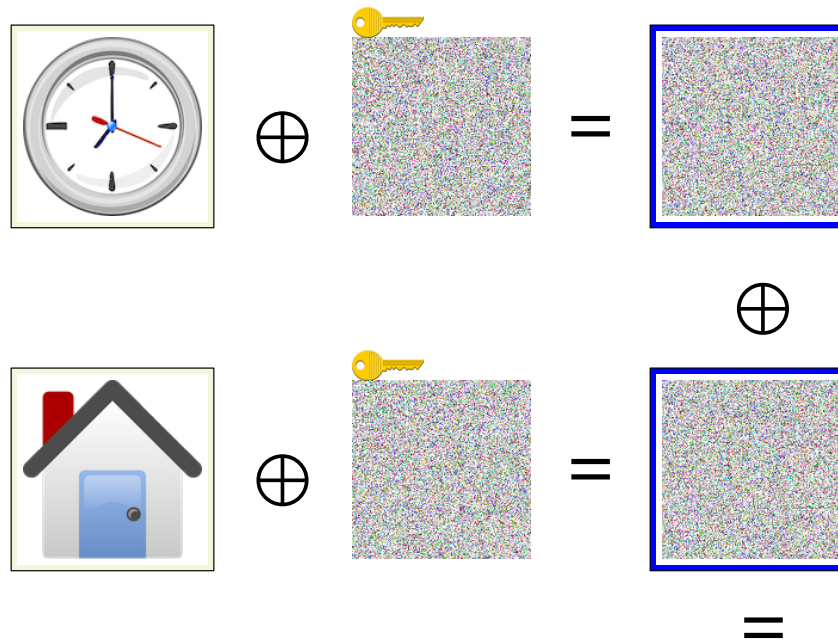
Misuse of Stream Cipher and Key



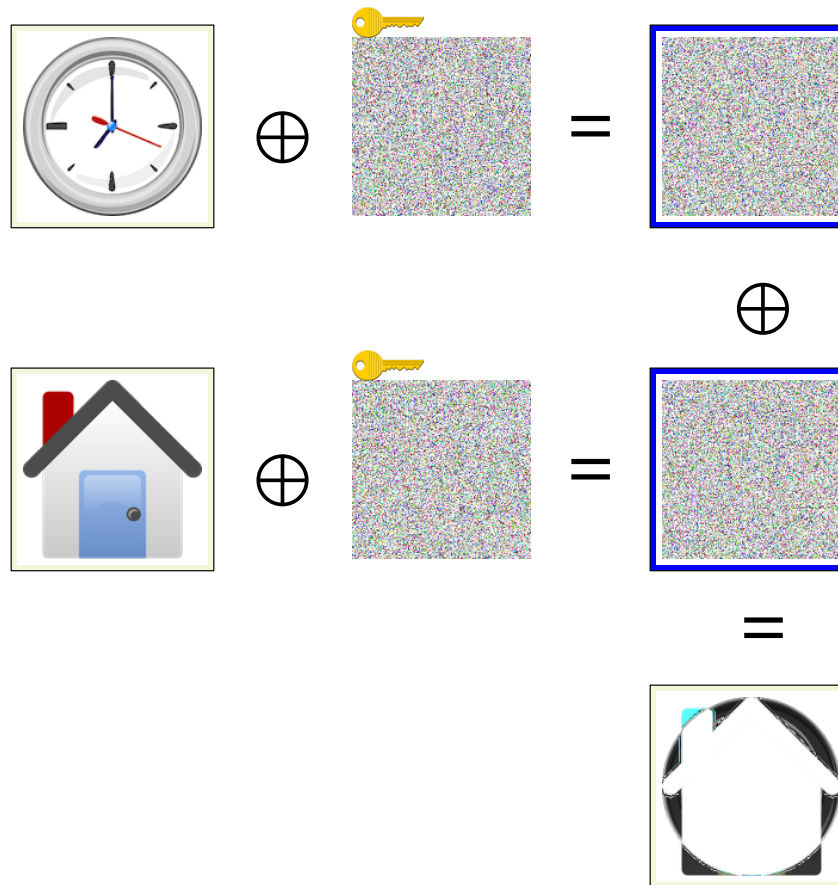
Misuse of Stream Cipher and Key



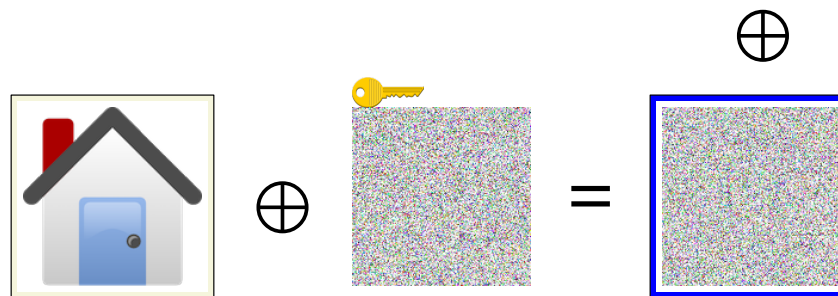
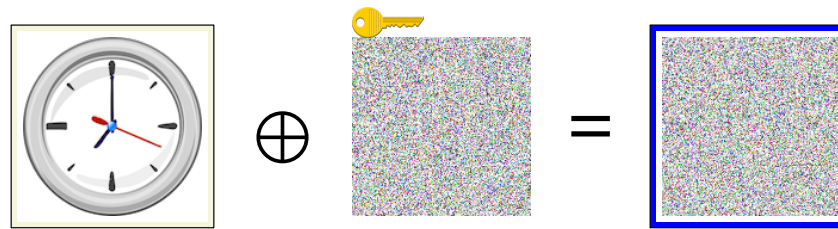
Misuse of Stream Cipher and Key



Misuse of Stream Cipher and Key



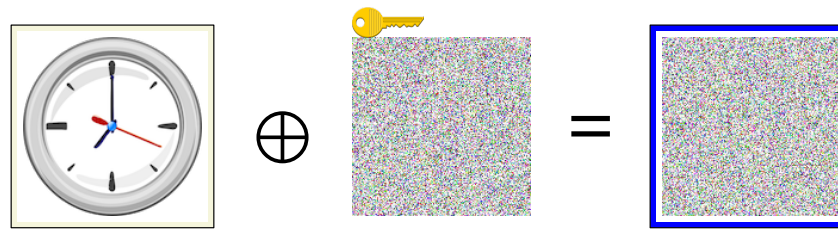
Misuse of Stream Cipher and Key



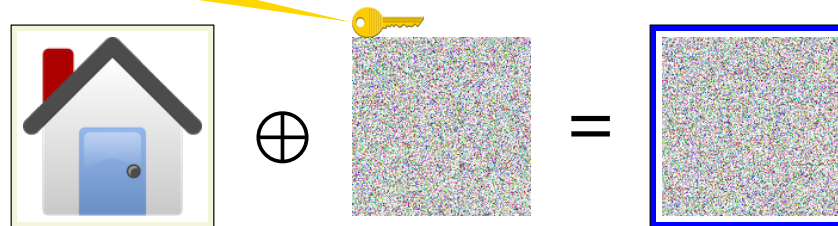
$$\begin{aligned}
 CT_a &= PT_a \oplus K \\
 CT_b &= PT_b \oplus K \\
 CT_a \oplus CT_b &= PT_a \oplus K \oplus PT_b \oplus K \\
 &= PT_a \oplus PT_b
 \end{aligned}$$



Misuse of Stream Cipher and Key



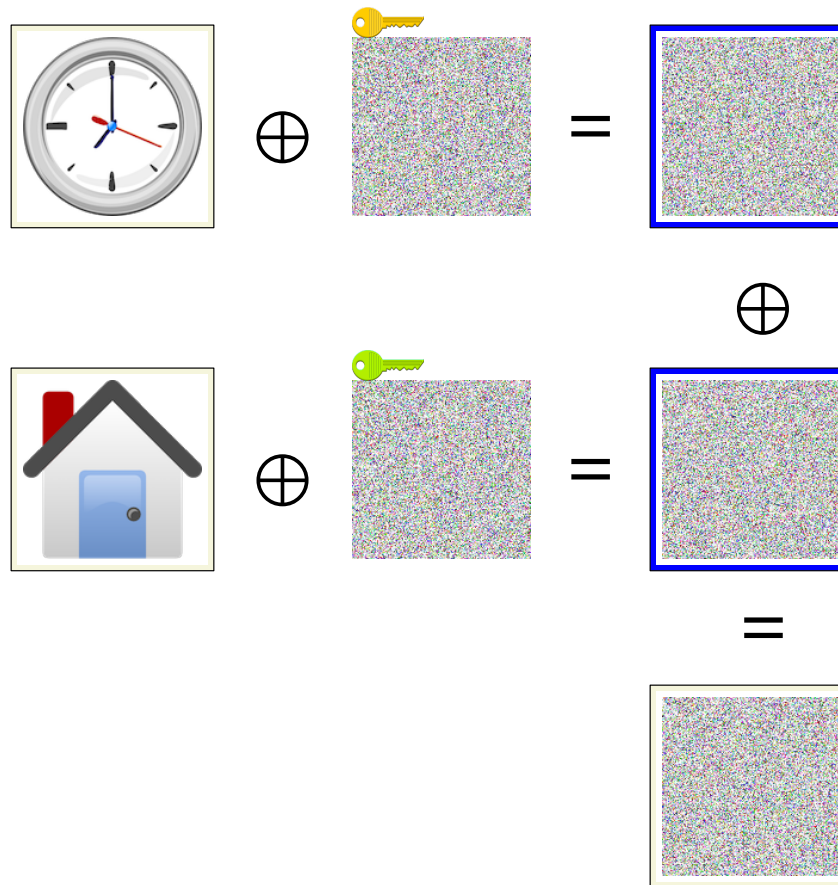
Don't use the same key twice!



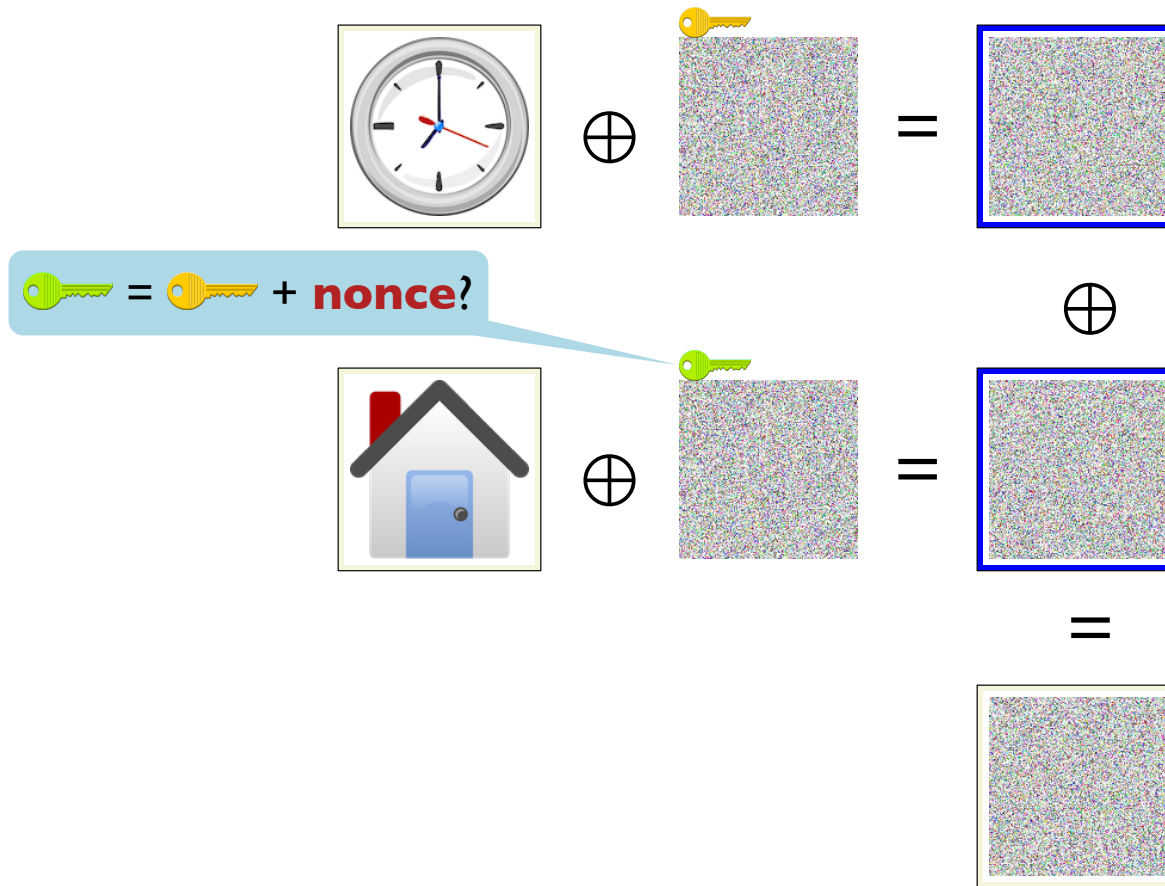
$$\begin{aligned}
 CT_a &= PT_a \oplus K \\
 CT_b &= PT_b \oplus K \\
 CT_a \oplus CT_b &= PT_a \oplus K \oplus PT_b \oplus K \\
 &= PT_a \oplus PT_b
 \end{aligned}$$



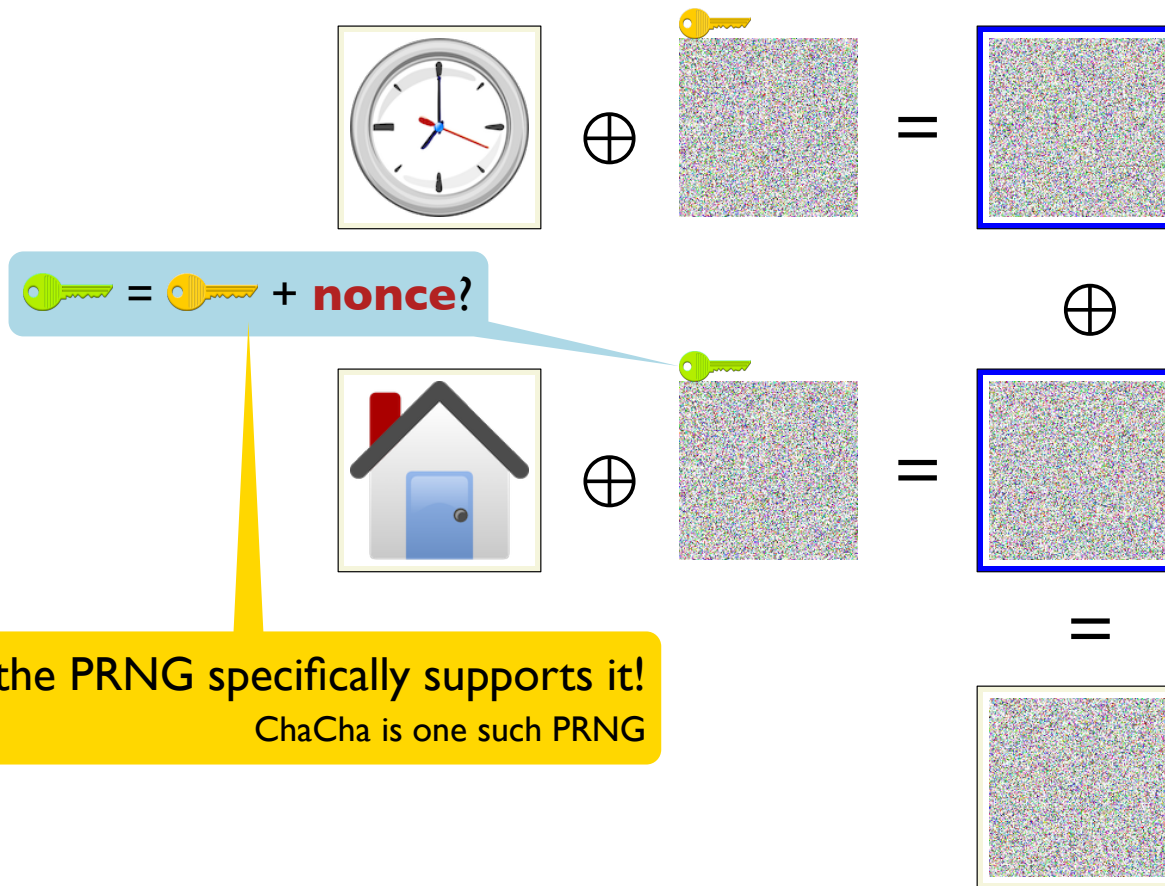
Potentially Correct Use of Stream Cipher and Key



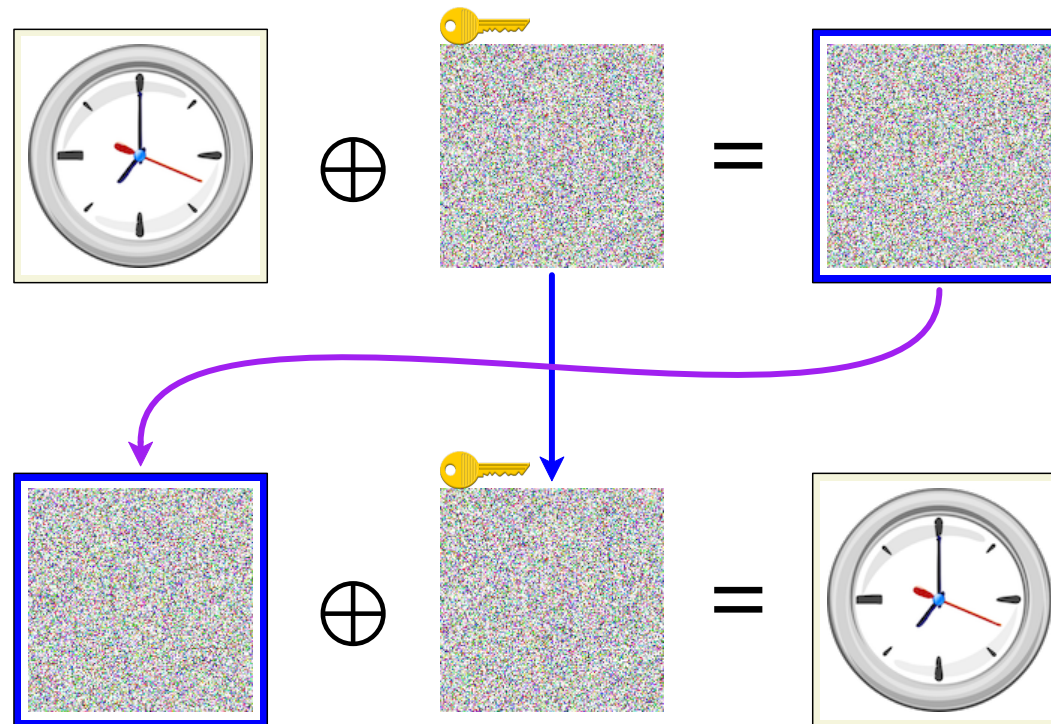
Potentially Correct Use of Stream Cipher and Key



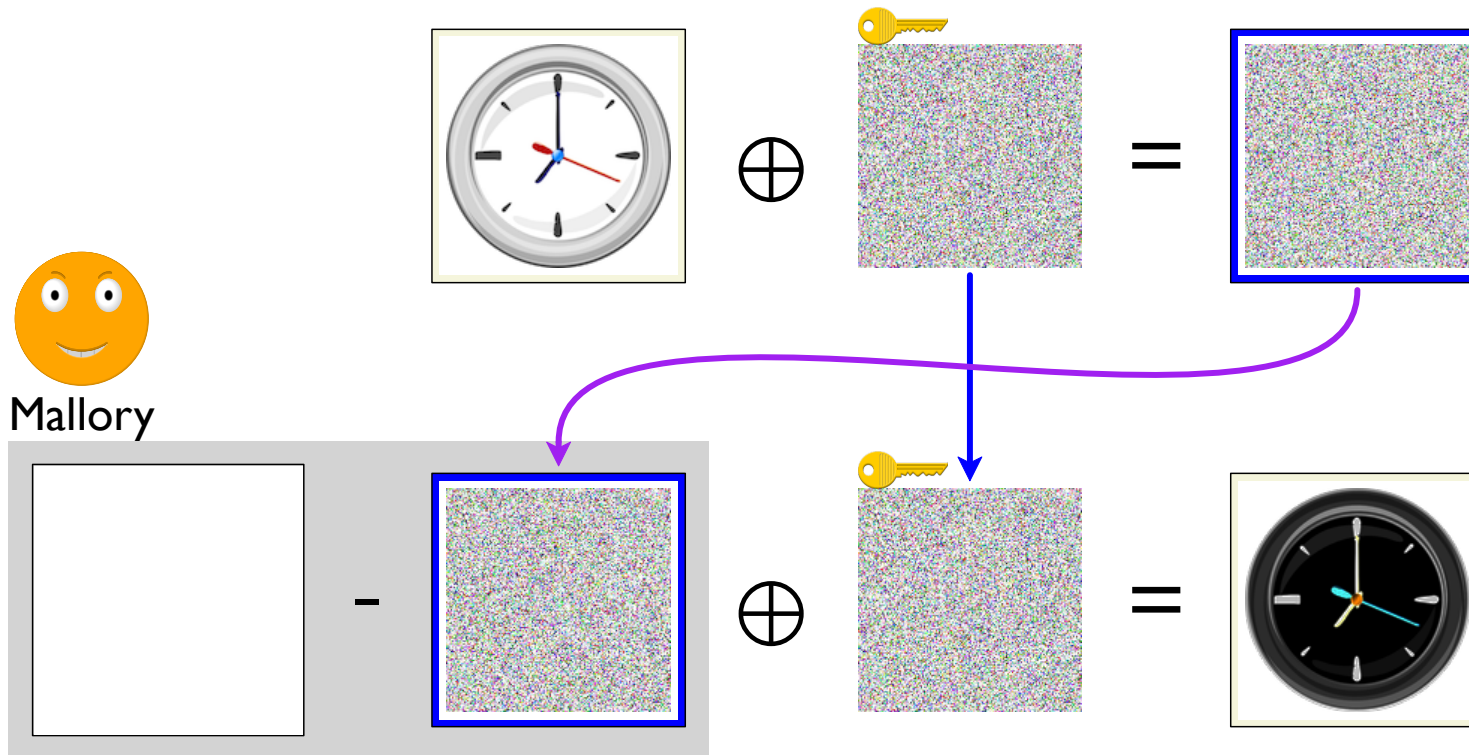
Potentially Correct Use of Stream Cipher and Key



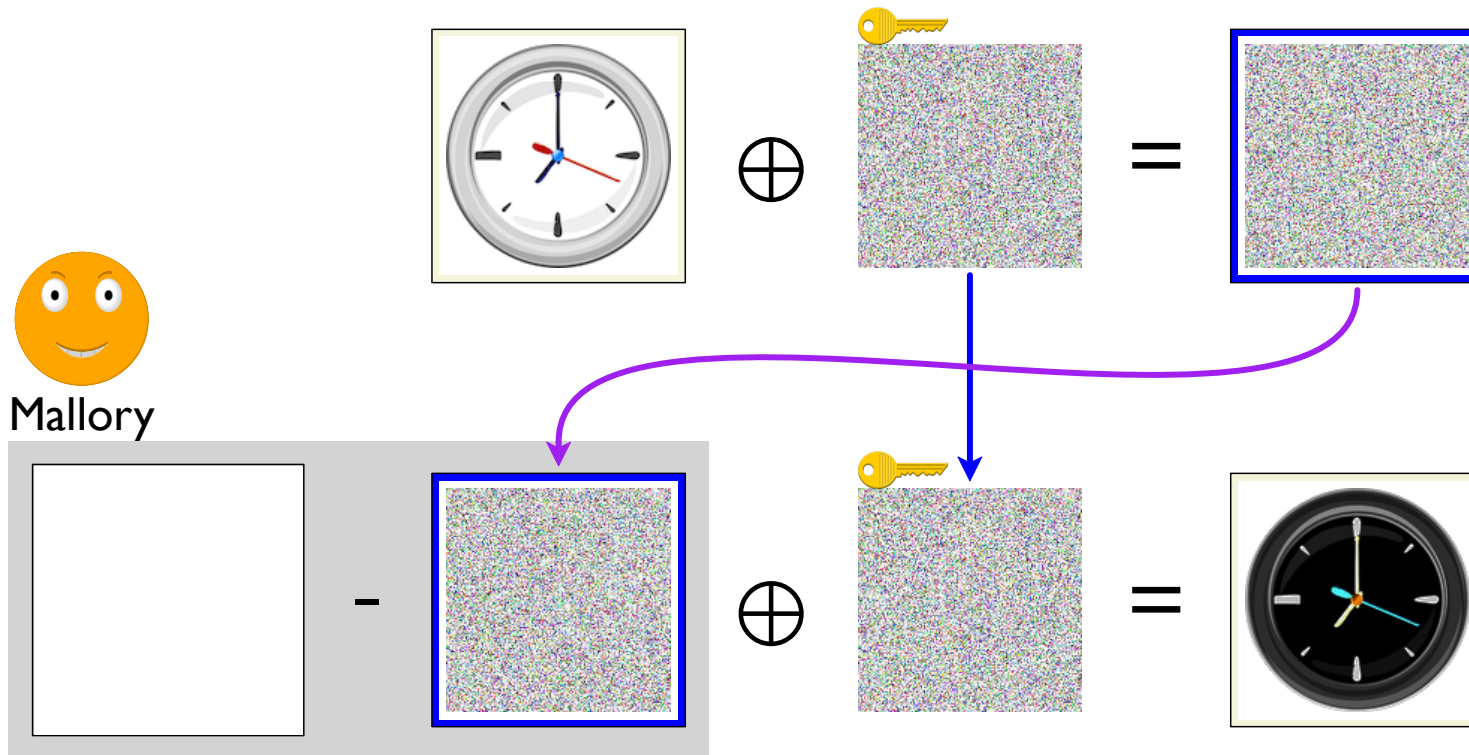
Stream Cipher Lack of Integrity



Stream Cipher Lack of Integrity



Stream Cipher Lack of Integrity



Again, we'll need more ingredients...

Creating a Really Good PRNG

We need a *cryptographically secure* PRNG (**CSPRNG**)

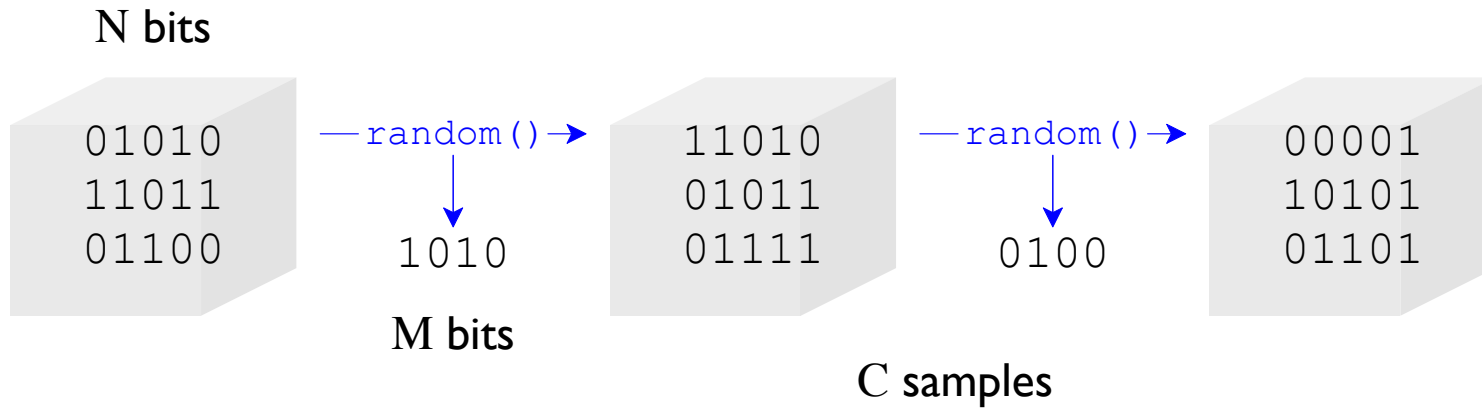
- Cannot predict next output from previous outputs
- Passes statistical randomness tests

Creating a Really Good PRNG

We need a *cryptographically secure* PRNG (**CSPRNG**)

- Cannot predict next output from previous outputs
- Passes statistical randomness tests
- Cannot use current state to predict *previous* output

PRNG

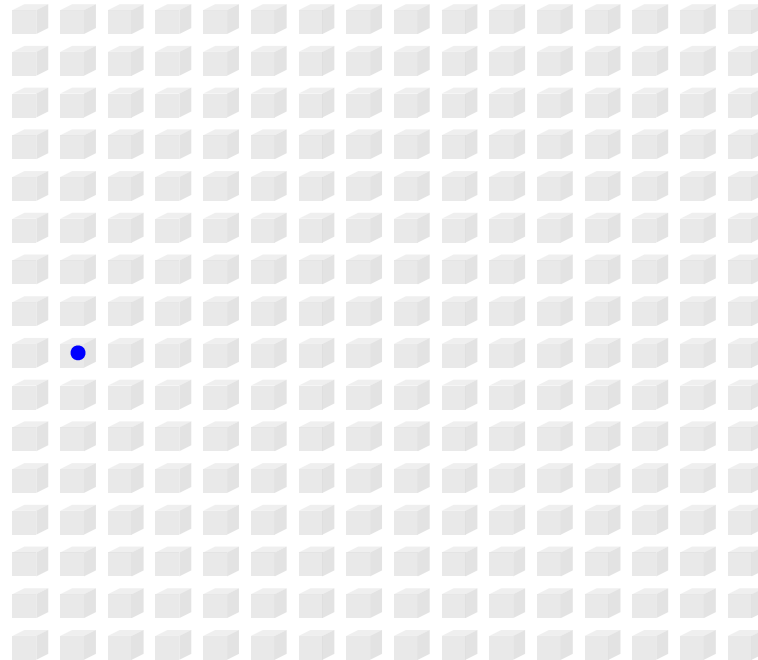


2^N possible states

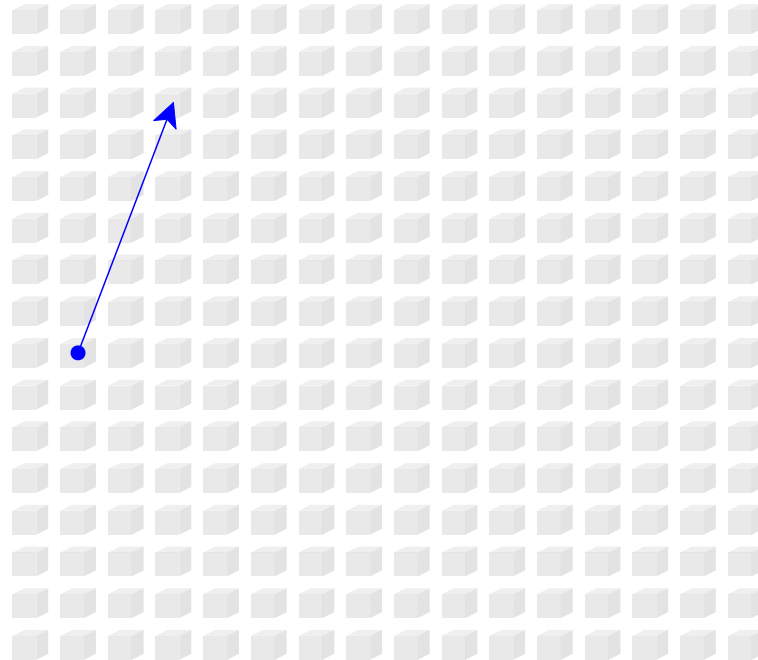
$2^{C \times M}$ possible sequences

⇒ lots of overlap when $C \times M \gg N$

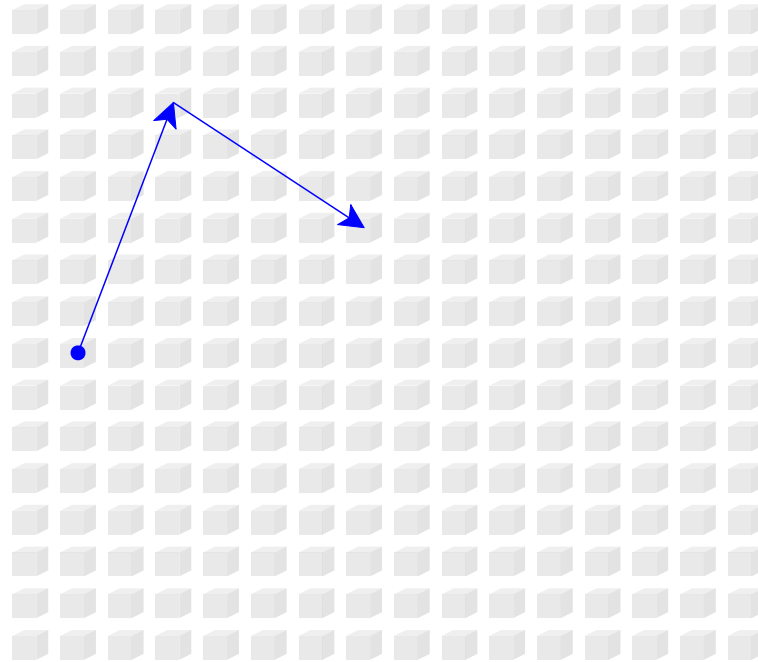
PRNG States



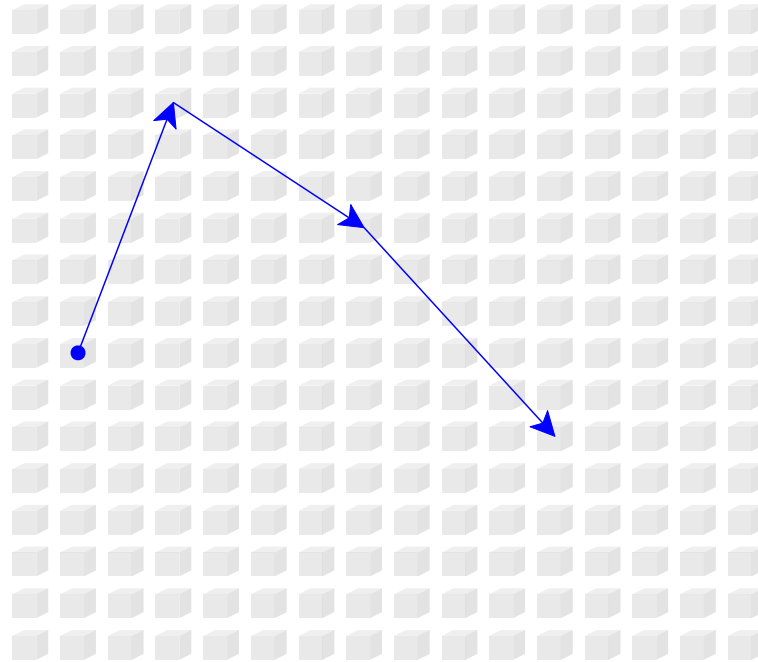
PRNG States



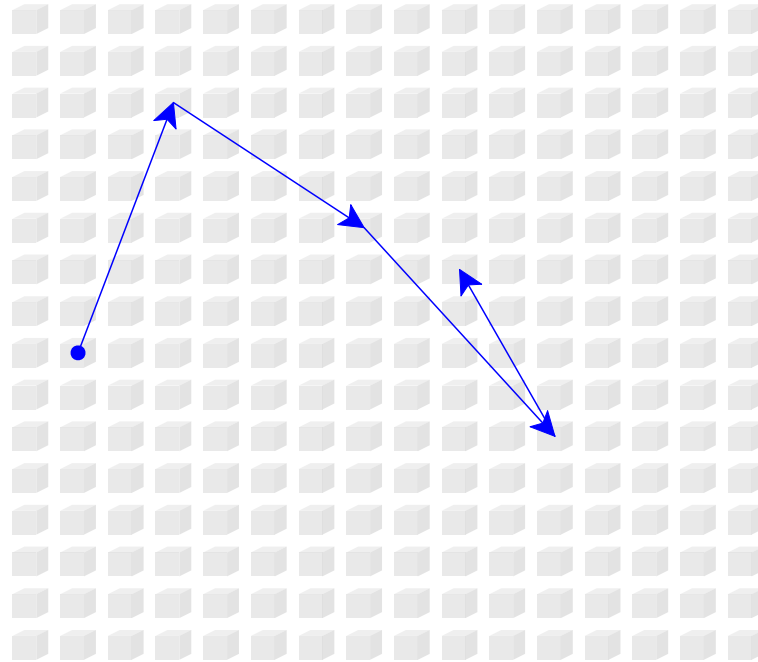
PRNG States



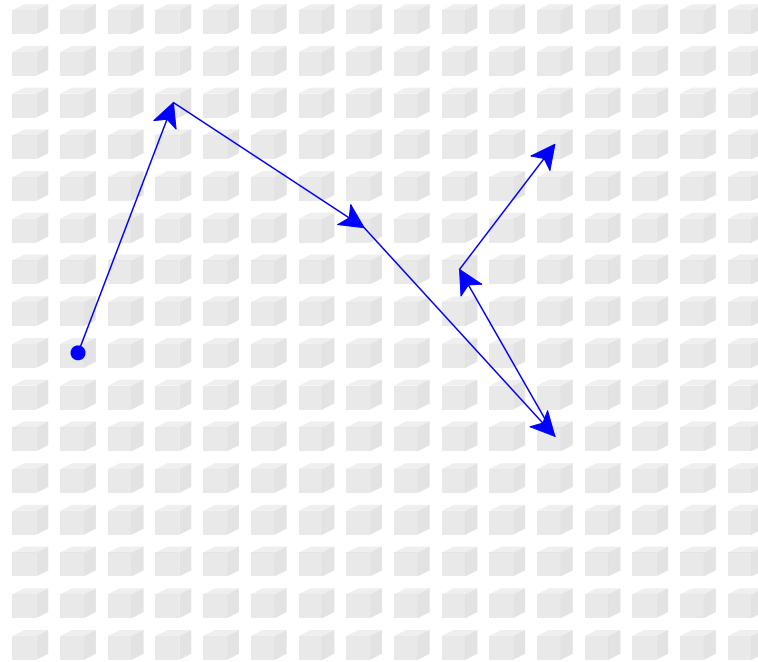
PRNG States



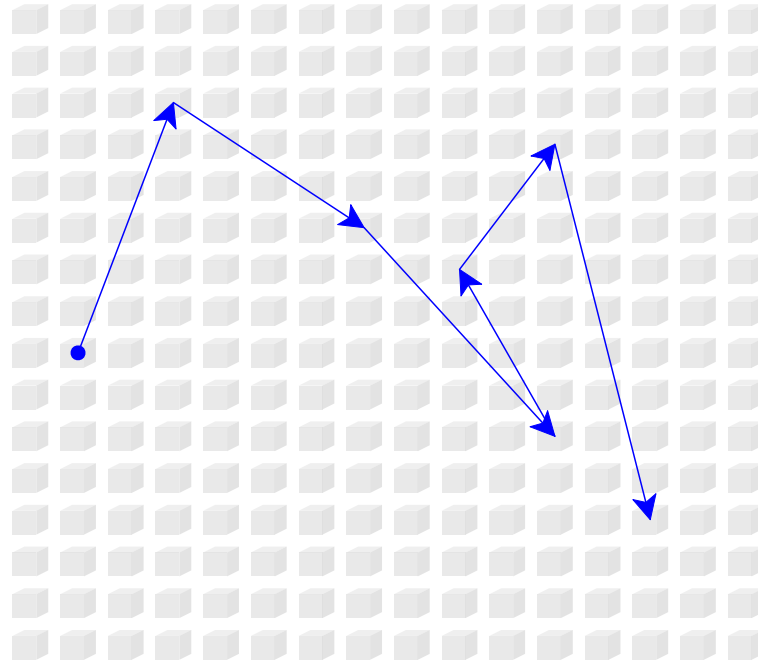
PRNG States



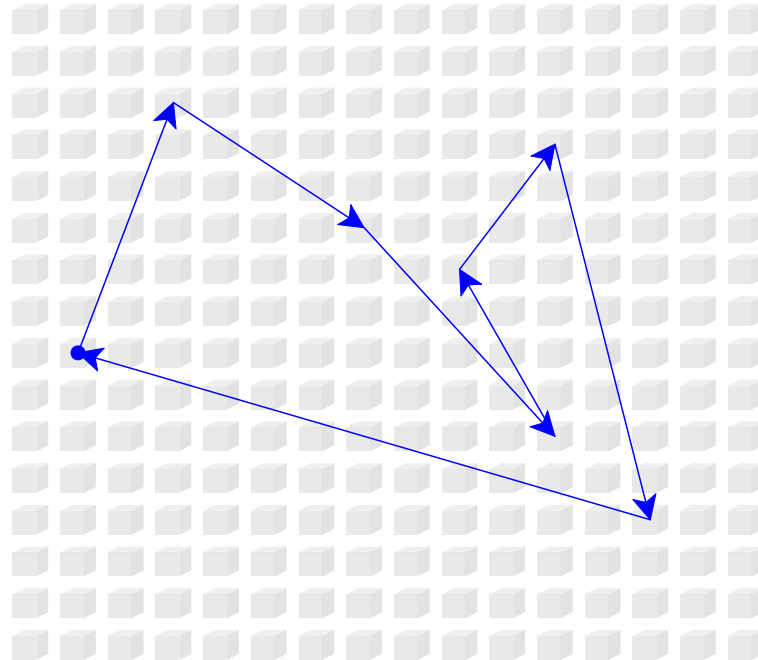
PRNG States



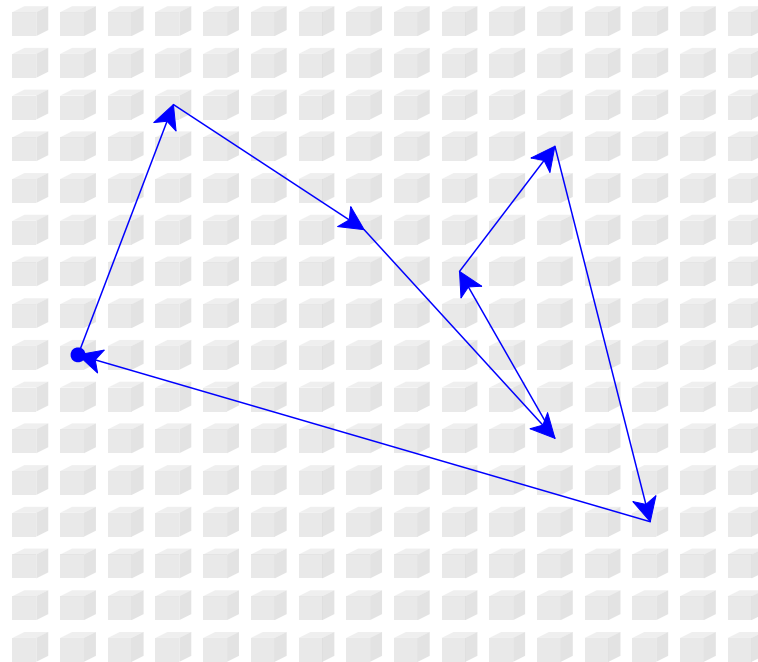
PRNG States



PRNG States

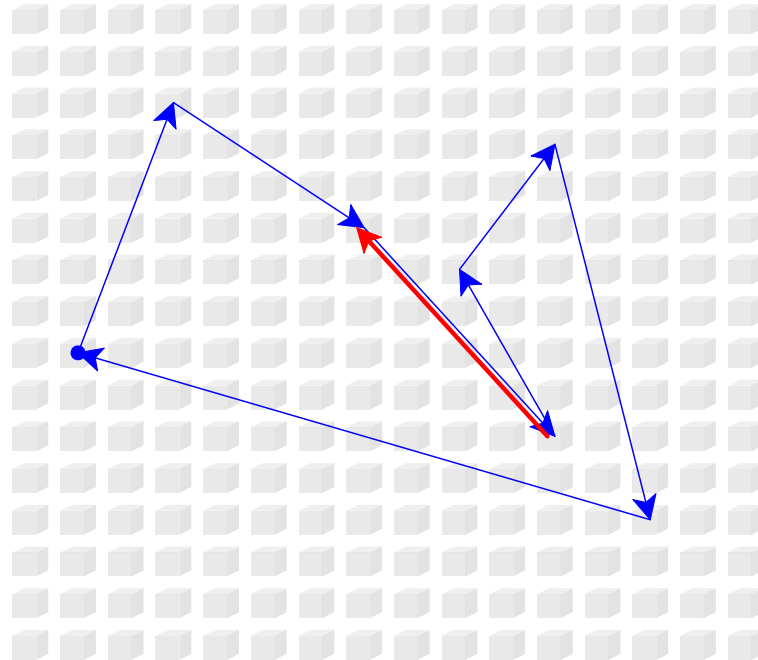


PRNG States



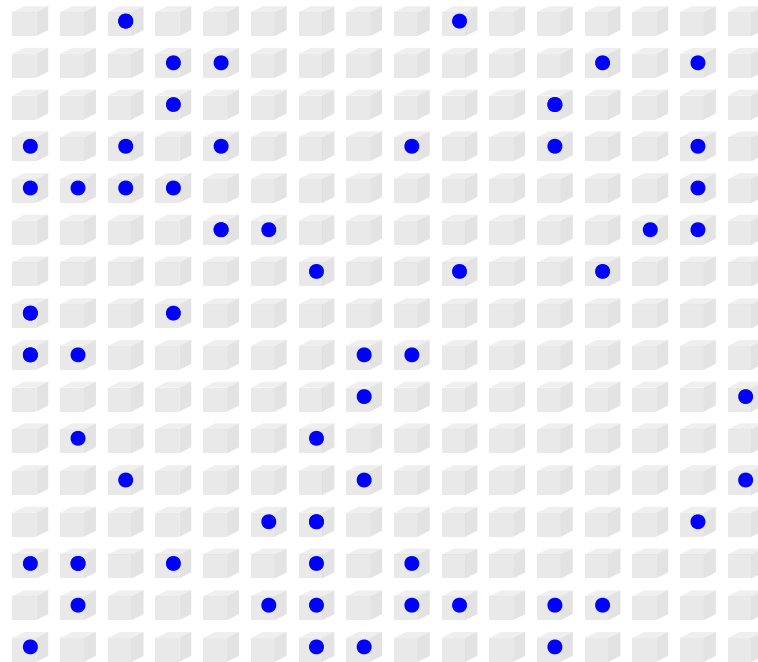
Cycle is inevitable, but we want it to be as long as possible

PRNG States



CSPRNG implies that
you can't reverse an arrow
even if you see the state

PRNG States



Key to initialize usually
< N bits, so only
some starts are possible

Historical PRNG: RC4

| | | | | | | | |
|---------------------------|-----|-----|-----|-----|-----|-----|-----|
| <i>i</i> = 0 <i>j</i> = 0 | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 |
| 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 |
| 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 |
| 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 |
| 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 |
| 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

RC4 is “Ron’s Code”

Used in the 1990s, but several vulnerabilities have been found, so don’t use it

Historical PRNG: RC4

| | | | | | | | |
|-----|-------------|-----|-----|-----|-----|-----|-----|
| | i = 0 j = 0 | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 |
| 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 |
| 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 |
| 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 |
| 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 |
| 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

```
def S = Array(0..255)
def i = 0
def j = 0
```

Historical PRNG: RC4

| | | | | | | | |
|-----|-------------|-----|-----|-----|-----|-----|-----|
| | i = 0 j = 0 | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 |
| 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 |
| 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 |
| 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 |
| 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 |
| 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

```
j := 0
for (i: 0..255):
    j := (j + S[i] + key[i % key.length]) % 256
    swap(S[i], S[j])

i := 0
j := 0
```

This and later code based on
<https://en.wikipedia.org/wiki/RC4>

Historical PRNG: RC4

```
      i = 0  j = 0
75  51 132 157 192 200 29 168
74 243 131 228 18 112 130 144
91 143 236 34 41 185 204 92
191 216 186 14 110 77 8 35
188 27 103 137 182 64 59 105
215 247 238 126 138 26 227 55
21 84 104 78 135 113 255 172
56 89 187 28 62 32 45 65
36 251 152 116 189 7 108 46
202 162 159 83 31 154 11 231
106 13 0 217 20 229 102 118
82 85 176 97 214 151 6 4
142 245 134 60 225 165 3 39
86 101 90 127 197 72 117 146
47 195 42 128 100 253 174 209
25 239 114 219 244 234 163 190
183 235 54 98 153 121 123 38
40 180 179 139 203 70 5 24
43 199 224 213 210 220 173 241
23 88 196 79 242 58 9 73
141 160 193 181 19 233 63 80
30 81 111 226 175 150 207 222
17 119 230 96 71 87 133 198
95 169 155 212 66 49 205 2
76 115 37 194 57 22 223 178
16 12 93 237 240 33 206 69
53 158 148 15 122 136 161 246
201 44 171 67 184 109 252 50
170 145 149 140 94 218 156 208
1 129 68 48 254 164 250 167
248 125 177 166 232 120 107 99
249 221 52 124 10 211 61 147
```

```
def key = "Secret"
```

```
j := 0
```

```
for (i: 0..255):
```

```
  j := (j + S[i] + key[i % key.length]) % 256
```

```
  swap(S[i], S[j])
```

```
i := 0
```

```
j := 0
```


Historical PRNG: RC4

```
i = 0 j = 0
75 51 132 157 192 200 29 168
74 243 131 228 18 112 130 144
91 143 236 34 41 185 204 92
191 216 186 14 110 77 8 35
188 27 103 137 182 64 59 105
215 247 238 126 138 26 227 55
21 84 104 78 135 113 255 172
56 89 187 28 62 32 45 65
36 251 152 116 189 7 108 46
202 162 159 83 31 154 11 231
106 13 0 217 20 229 102 118
82 85 176 97 214 151 6 4
142 245 134 60 225 165 3 39
86 101 90 127 197 72 117 146
47 195 42 128 100 253 174 209
25 239 114 219 244 234 163 190
183 235 54 98 153 121 123 38
40 180 179 139 203 70 5 24
43 199 224 213 210 220 173 241
23 88 196 79 242 58 9 73
141 160 193 181 19 233 63 80
30 81 111 226 175 150 207 222
17 119 230 96 71 87 133 198
95 169 155 212 66 49 205 2
76 115 37 194 57 22 223 178
16 12 93 237 240 33 206 69
53 158 148 15 122 136 161 246
201 44 171 67 184 109 252 50
170 145 149 140 94 218 156 208
1 129 68 48 254 164 250 167
248 125 177 166 232 120 107 99
249 221 52 124 10 211 61 147
```

any length, but between 5 and 16 bytes is common

```
def key = "Secret"
```

```
j := 0
```

```
for (i: 0..255):
```

```
    j := (j + S[i] + key[i % key.length]) % 256
```

```
    swap(S[i], S[j])
```

```
i := 0
```

```
j := 0
```

Historical PRNG: RC4

```
      i = 0  j = 0
75  51 132 157 192 200 29 168
74 243 131 228 18 112 130 144
91 143 236 34 41 185 204 92
191 216 186 14 110 77 8 35
188 27 103 137 182 64 59 105
215 247 238 126 138 26 227 55
21 84 104 78 135 113 255 172
56 89 187 28 62 32 45 65
36 251 152 116 189 7 108 46
202 162 159 83 31 154 11 231
106 13 0 217 20 229 102 118
82 85 176 97 214 151 6 4
142 245 134 60 225 165 3 39
86 101 90 127 197 72 117 146
47 195 42 128 100 253 174 209
25 239 114 219 244 234 163 190
183 235 54 98 153 121 123 38
40 180 179 139 203 70 5 24
43 199 224 213 210 220 173 241
23 88 196 79 242 58 9 73
141 160 193 181 19 233 63 80
30 81 111 226 175 150 207 222
17 119 230 96 71 87 133 198
95 169 155 212 66 49 205 2
76 115 37 194 57 22 223 178
16 12 93 237 240 33 206 69
53 158 148 15 122 136 161 246
201 44 171 67 184 109 252 50
170 145 149 140 94 218 156 208
1 129 68 48 254 164 250 167
248 125 177 166 232 120 107 99
249 221 52 124 10 211 61 147
```

```
def next_byte():
    i := (i+1) mod 256
    j := (j+S[i]) mod 256
    swap(S[i], S[j])
    return S[(S[i] + S[j]) mod 256]
```

Historical PRNG: RC4

`i = 1 j = 51`

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 75 | 78 | 132 | 157 | 192 | 200 | 29 | 168 |
| 74 | 243 | 131 | 228 | 18 | 112 | 130 | 144 |
| 91 | 143 | 236 | 34 | 41 | 185 | 204 | 92 |
| 191 | 216 | 186 | 14 | 110 | 77 | 8 | 35 |
| 188 | 27 | 103 | 137 | 182 | 64 | 59 | 105 |
| 215 | 247 | 238 | 126 | 138 | 26 | 227 | 55 |
| 21 | 84 | 104 | 51 | 135 | 113 | 255 | 172 |
| 56 | 89 | 187 | 28 | 62 | 32 | 45 | 65 |
| 36 | 251 | 152 | 116 | 189 | 7 | 108 | 46 |
| 202 | 162 | 159 | 83 | 31 | 154 | 11 | 231 |
| 106 | 13 | 0 | 217 | 20 | 229 | 102 | 118 |
| 82 | 85 | 176 | 97 | 214 | 151 | 6 | 4 |
| 142 | 245 | 134 | 60 | 225 | 165 | 3 | 39 |
| 86 | 101 | 90 | 127 | 197 | 72 | 117 | 146 |
| 47 | 195 | 42 | 128 | 100 | 253 | 174 | 209 |
| 25 | 239 | 114 | 219 | 244 | 234 | 163 | 190 |
| 183 | 235 | 54 | 98 | 153 | 121 | 123 | 38 |
| 40 | 180 | 179 | 139 | 203 | 70 | 5 | 24 |
| 43 | 199 | 224 | 213 | 210 | 220 | 173 | 241 |
| 23 | 88 | 196 | 79 | 242 | 58 | 9 | 73 |
| 141 | 160 | 193 | 181 | 19 | 233 | 63 | 80 |
| 30 | 81 | 111 | 226 | 175 | 150 | 207 | 222 |
| 17 | 119 | 230 | 96 | 71 | 87 | 133 | 198 |
| 95 | 169 | 155 | 212 | 66 | 49 | 205 | 2 |
| 76 | 115 | 37 | 194 | 57 | 22 | 223 | 178 |
| 16 | 12 | 93 | 237 | 240 | 33 | 206 | 69 |
| 53 | 158 | 148 | 15 | 122 | 136 | 161 | 246 |
| 201 | 44 | 171 | 67 | 184 | 109 | 252 | 50 |
| 170 | 145 | 149 | 140 | 94 | 218 | 156 | 208 |
| 1 | 129 | 68 | 48 | 254 | 164 | 250 | 167 |
| 248 | 125 | 177 | 166 | 232 | 120 | 107 | 99 |
| 249 | 221 | 52 | 124 | 10 | 211 | 61 | 147 |

```
def next_byte():  
    i := (i+1) mod 256  
    j := (j+S[i]) mod 256  
    swap(S[i], S[j])  
    return S[(S[i] + S[j]) mod 256]
```

```
next_byte() = 235
```

Historical PRNG: RC4

$i = 2$ $j = 183$

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 75 | 78 | 198 | 157 | 192 | 200 | 29 | 168 |
| 74 | 243 | 131 | 228 | 18 | 112 | 130 | 144 |
| 91 | 143 | 236 | 34 | 41 | 185 | 204 | 92 |
| 191 | 216 | 186 | 14 | 110 | 77 | 8 | 35 |
| 188 | 27 | 103 | 137 | 182 | 64 | 59 | 105 |
| 215 | 247 | 238 | 126 | 138 | 26 | 227 | 55 |
| 21 | 84 | 104 | 51 | 135 | 113 | 255 | 172 |
| 56 | 89 | 187 | 28 | 62 | 32 | 45 | 65 |
| 36 | 251 | 152 | 116 | 189 | 7 | 108 | 46 |
| 202 | 162 | 159 | 83 | 31 | 154 | 11 | 231 |
| 106 | 13 | 0 | 217 | 20 | 229 | 102 | 118 |
| 82 | 85 | 176 | 97 | 214 | 151 | 6 | 4 |
| 142 | 245 | 134 | 60 | 225 | 165 | 3 | 39 |
| 86 | 101 | 90 | 127 | 197 | 72 | 117 | 146 |
| 47 | 195 | 42 | 128 | 100 | 253 | 174 | 209 |
| 25 | 239 | 114 | 219 | 244 | 234 | 163 | 190 |
| 183 | 235 | 54 | 98 | 153 | 121 | 123 | 38 |
| 40 | 180 | 179 | 139 | 203 | 70 | 5 | 24 |
| 43 | 199 | 224 | 213 | 210 | 220 | 173 | 241 |
| 23 | 88 | 196 | 79 | 242 | 58 | 9 | 73 |
| 141 | 160 | 193 | 181 | 19 | 233 | 63 | 80 |
| 30 | 81 | 111 | 226 | 175 | 150 | 207 | 222 |
| 17 | 119 | 230 | 96 | 71 | 87 | 133 | 132 |
| 95 | 169 | 155 | 212 | 66 | 49 | 205 | 2 |
| 76 | 115 | 37 | 194 | 57 | 22 | 223 | 178 |
| 16 | 12 | 93 | 237 | 240 | 33 | 206 | 69 |
| 53 | 158 | 148 | 15 | 122 | 136 | 161 | 246 |
| 201 | 44 | 171 | 67 | 184 | 109 | 252 | 50 |
| 170 | 145 | 149 | 140 | 94 | 218 | 156 | 208 |
| 1 | 129 | 68 | 48 | 254 | 164 | 250 | 167 |
| 248 | 125 | 177 | 166 | 232 | 120 | 107 | 99 |
| 249 | 221 | 52 | 124 | 10 | 211 | 61 | 147 |

```
def next_byte():  
    i := (i+1) mod 256  
    j := (j+S[i]) mod 256  
    swap(S[i], S[j])  
    return S[(S[i] + S[j]) mod 256]
```

```
next_byte() = 159
```

Historical PRNG: RC4

| | | | | | | | |
|----------------------|-----|-----|-----|-----|-----|-----|-----|
| $i = 3 \quad j = 84$ | | | | | | | |
| 75 | 78 | 198 | 20 | 192 | 200 | 29 | 168 |
| 74 | 243 | 131 | 228 | 18 | 112 | 130 | 144 |
| 91 | 143 | 236 | 34 | 41 | 185 | 204 | 92 |
| 191 | 216 | 186 | 14 | 110 | 77 | 8 | 35 |
| 188 | 27 | 103 | 137 | 182 | 64 | 59 | 105 |
| 215 | 247 | 238 | 126 | 138 | 26 | 227 | 55 |
| 21 | 84 | 104 | 51 | 135 | 113 | 255 | 172 |
| 56 | 89 | 187 | 28 | 62 | 32 | 45 | 65 |
| 36 | 251 | 152 | 116 | 189 | 7 | 108 | 46 |
| 202 | 162 | 159 | 83 | 31 | 154 | 11 | 231 |
| 106 | 13 | 0 | 217 | 157 | 229 | 102 | 118 |
| 82 | 85 | 176 | 97 | 214 | 151 | 6 | 4 |
| 142 | 245 | 134 | 60 | 225 | 165 | 3 | 39 |
| 86 | 101 | 90 | 127 | 197 | 72 | 117 | 146 |
| 47 | 195 | 42 | 128 | 100 | 253 | 174 | 209 |
| 25 | 239 | 114 | 219 | 244 | 234 | 163 | 190 |
| 183 | 235 | 54 | 98 | 153 | 121 | 123 | 38 |
| 40 | 180 | 179 | 139 | 203 | 70 | 5 | 24 |
| 43 | 199 | 224 | 213 | 210 | 220 | 173 | 241 |
| 23 | 88 | 196 | 79 | 242 | 58 | 9 | 73 |
| 141 | 160 | 193 | 181 | 19 | 233 | 63 | 80 |
| 30 | 81 | 111 | 226 | 175 | 150 | 207 | 222 |
| 17 | 119 | 230 | 96 | 71 | 87 | 133 | 132 |
| 95 | 169 | 155 | 212 | 66 | 49 | 205 | 2 |
| 76 | 115 | 37 | 194 | 57 | 22 | 223 | 178 |
| 16 | 12 | 93 | 237 | 240 | 33 | 206 | 69 |
| 53 | 158 | 148 | 15 | 122 | 136 | 161 | 246 |
| 201 | 44 | 171 | 67 | 184 | 109 | 252 | 50 |
| 170 | 145 | 149 | 140 | 94 | 218 | 156 | 208 |
| 1 | 129 | 68 | 48 | 254 | 164 | 250 | 167 |
| 248 | 125 | 177 | 166 | 232 | 120 | 107 | 99 |
| 249 | 221 | 52 | 124 | 10 | 211 | 61 | 147 |

```
def next_byte():  
    i := (i+1) mod 256  
    j := (j+S[i]) mod 256  
    swap(S[i], S[j])  
    return S[(S[i] + S[j]) mod 256]
```

```
next_byte() = 119
```

Historical PRNG: RC4

| | | | | | | | |
|------------------|-----|-----|-----|-----|-----|-----|-----|
| $i = 4$ $j = 20$ | | | | | | | |
| 75 | 78 | 198 | 20 | 41 | 200 | 29 | 168 |
| 74 | 243 | 131 | 228 | 18 | 112 | 130 | 144 |
| 91 | 143 | 236 | 34 | 192 | 185 | 204 | 92 |
| 191 | 216 | 186 | 14 | 110 | 77 | 8 | 35 |
| 188 | 27 | 103 | 137 | 182 | 64 | 59 | 105 |
| 215 | 247 | 238 | 126 | 138 | 26 | 227 | 55 |
| 21 | 84 | 104 | 51 | 135 | 113 | 255 | 172 |
| 56 | 89 | 187 | 28 | 62 | 32 | 45 | 65 |
| 36 | 251 | 152 | 116 | 189 | 7 | 108 | 46 |
| 202 | 162 | 159 | 83 | 31 | 154 | 11 | 231 |
| 106 | 13 | 0 | 217 | 157 | 229 | 102 | 118 |
| 82 | 85 | 176 | 97 | 214 | 151 | 6 | 4 |
| 142 | 245 | 134 | 60 | 225 | 165 | 3 | 39 |
| 86 | 101 | 90 | 127 | 197 | 72 | 117 | 146 |
| 47 | 195 | 42 | 128 | 100 | 253 | 174 | 209 |
| 25 | 239 | 114 | 219 | 244 | 234 | 163 | 190 |
| 183 | 235 | 54 | 98 | 153 | 121 | 123 | 38 |
| 40 | 180 | 179 | 139 | 203 | 70 | 5 | 24 |
| 43 | 199 | 224 | 213 | 210 | 220 | 173 | 241 |
| 23 | 88 | 196 | 79 | 242 | 58 | 9 | 73 |
| 141 | 160 | 193 | 181 | 19 | 233 | 63 | 80 |
| 30 | 81 | 111 | 226 | 175 | 150 | 207 | 222 |
| 17 | 119 | 230 | 96 | 71 | 87 | 133 | 132 |
| 95 | 169 | 155 | 212 | 66 | 49 | 205 | 2 |
| 76 | 115 | 37 | 194 | 57 | 22 | 223 | 178 |
| 16 | 12 | 93 | 237 | 240 | 33 | 206 | 69 |
| 53 | 158 | 148 | 15 | 122 | 136 | 161 | 246 |
| 201 | 44 | 171 | 67 | 184 | 109 | 252 | 50 |
| 170 | 145 | 149 | 140 | 94 | 218 | 156 | 208 |
| 1 | 129 | 68 | 48 | 254 | 164 | 250 | 167 |
| 248 | 125 | 177 | 166 | 232 | 120 | 107 | 99 |
| 249 | 221 | 52 | 124 | 10 | 211 | 61 | 147 |

```
def next_byte():  
    i := (i+1) mod 256  
    j := (j+S[i]) mod 256  
    swap(S[i], S[j])  
    return S[(S[i] + S[j]) mod 256]
```

```
next_byte() = 129
```

Historical PRNG: RC4

i = 4 *j* = 20

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 75 | 78 | 198 | 20 | 41 | 200 | 29 | 168 |
| 74 | 243 | 131 | 228 | 18 | 112 | 130 | 144 |
| 91 | 143 | 236 | 34 | 192 | 185 | 204 | 92 |
| 191 | 216 | 186 | 14 | 110 | 77 | 8 | 35 |
| 188 | 27 | 103 | 137 | 182 | 64 | 59 | 105 |
| 215 | 247 | 238 | 126 | 138 | 26 | 227 | 55 |
| 21 | 84 | 104 | 51 | 135 | 113 | 255 | 172 |
| 56 | 89 | 187 | 28 | 62 | 32 | 45 | 65 |
| 36 | 251 | 152 | 116 | 189 | 7 | 108 | 46 |
| 202 | 162 | 159 | 83 | 31 | 154 | 11 | 231 |
| 106 | 13 | 0 | 217 | 157 | 229 | 102 | 118 |
| 82 | 85 | 176 | 97 | 214 | 151 | 6 | 4 |
| 142 | 245 | 134 | 60 | 225 | 165 | 3 | 39 |
| 86 | 101 | 90 | 127 | 197 | 72 | 117 | 146 |
| 47 | 195 | 42 | 128 | 100 | 253 | 174 | 209 |
| 25 | 239 | 114 | 219 | 244 | 234 | 163 | 190 |
| 183 | 235 | 54 | 98 | 153 | 121 | 123 | 38 |
| 40 | 180 | 179 | 139 | 203 | 70 | 5 | 24 |
| 43 | 199 | 224 | 213 | 210 | 220 | 173 | 241 |
| 23 | 88 | 196 | 79 | 242 | 58 | 9 | 73 |
| 141 | 160 | 193 | 181 | 19 | 233 | 63 | 80 |
| 30 | 81 | 111 | 226 | 175 | 150 | 207 | 222 |
| 17 | 119 | 230 | 96 | 71 | 87 | 133 | 132 |
| 95 | 169 | 155 | 212 | 66 | 49 | 205 | 2 |
| 76 | 115 | 37 | 194 | 57 | 22 | 223 | 178 |
| 16 | 12 | 93 | 237 | 240 | 33 | 206 | 69 |
| 53 | 158 | 148 | 15 | 122 | 136 | 161 | 246 |
| 201 | 44 | 171 | 67 | 184 | 109 | 252 | 50 |
| 170 | 145 | 149 | 140 | 94 | 218 | 156 | 208 |
| 1 | 129 | 68 | 48 | 254 | 164 | 250 | 167 |
| 248 | 125 | 177 | 166 | 232 | 120 | 107 | 99 |
| 249 | 221 | 52 | 124 | 10 | 211 | 61 | 147 |

Vulnerabilities include

- first few bytes expose key
- bias in consecutive pairs
- bias toward repeating pairs

Modern PRNG: ChaCha

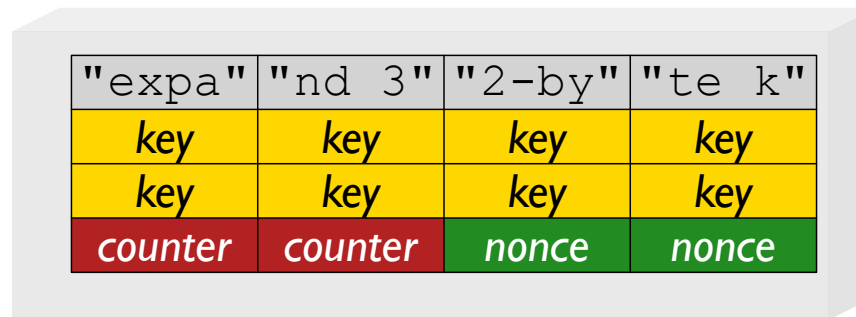
`chacha(key, counter, nonce) → PRNG`

| | | | |
|---------|---------|--------|--------|
| "expa" | "nd 3" | "2-by" | "te k" |
| key | key | key | key |
| key | key | key | key |
| counter | counter | nonce | nonce |

<https://en.wikipedia.org/wiki/Salsa20>

Modern PRNG: ChaCha

`chacha(key, counter, nonce) → PRNG`



| | | | |
|---------|---------|--------|--------|
| "expa" | "nd 3" | "2-by" | "te k" |
| key | key | key | key |
| key | key | key | key |
| counter | counter | nonce | nonce |

<https://en.wikipedia.org/wiki/Salsa20>

Applied alternately to columns then diagonals:

```
quarter_round(a, b, c, d):  
    a += b; d ⊕= a; d <<<= 16  
    c += d; b ⊕= c; b <<<= 12  
    a += b; d ⊕= a; d <<<= 8  
    c += d; b ⊕= c; b <<<= 7
```

Summary

Stream ciphers use a PRNG plus \oplus

Don't reuse a single key

Confidentiality only; need more layers for other goals

Modern CSPRNGs use a mixture of \lll and \oplus on internal state