Extending the Hill Cipher

Two secure modifications of the classic cipher

John Chase Matt Davis

Cryptography 625.480

December 2, 2010 / Final Paper Presentation

Outline

Introduction

Hill Cipher

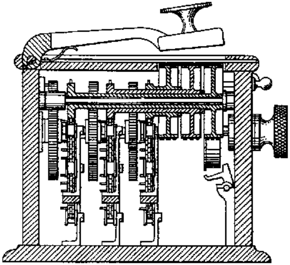
- Matrix encryption scheme introduced by Lester Hill in a 1929 paper.
- Machine built, but never used.
- Encryption scheme

$$Y = XK \pmod{m}$$

- Easily succumbs to known-plaintext attack. The key is recovered as

$$K = X^{-1}Y$$

- A dozen secure modifications. We'll look at two.



Introduction

Hill Cipher

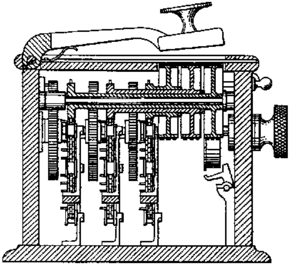
- Matrix encryption scheme introduced by Lester Hill in a 1929 paper.
- Machine built, but never used.
- Encryption scheme

$$Y = XK \pmod{m}$$

- Easily succumbs to known-plaintext attack. The key is recovered as

$$K = X^{-1}Y$$

- A dozen secure modifications. We'll look at two.



Introduction

Hill Cipher

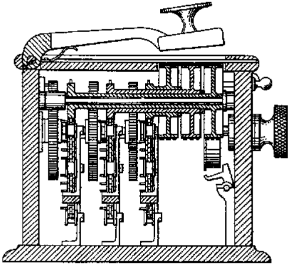
- Matrix encryption scheme introduced by Lester Hill in a 1929 paper.
- Machine built, but never used.
- Encryption scheme

$$Y = XK \pmod{m}$$

- Easily succumbs to known-plaintext attack. The key is recovered as

$$K = X^{-1}Y$$

- A dozen secure modifications. We'll look at two.



Introduction

Hill Cipher

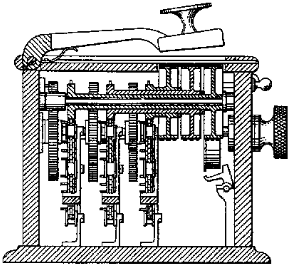
- Matrix encryption scheme introduced by Lester Hill in a 1929 paper.
- Machine built, but never used.
- Encryption scheme

$$Y = XK \pmod{m}$$

- Easily succumbs to known-plaintext attack. The key is recovered as

$$K = X^{-1}Y$$

- A dozen secure modifications. We'll look at two.



Introduction

Hill Cipher

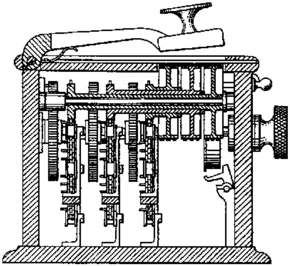
- Matrix encryption scheme introduced by Lester Hill in a 1929 paper.
- Machine built, but never used.
- Encryption scheme

$$Y = XK \pmod{m}$$

- Easily succumbs to known-plaintext attack. The key is recovered as

$$K = X^{-1}Y$$

- A dozen secure modifications. We'll look at two.



SVK Hill Cipher

Sastry, Varanasi, and Kumar (2010).

Key Features

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Encoded plaintext must be loaded into square matrices having the same size as the keys.
- Multiple iterations.

SVK Hill Cipher

Sastry, Varanasi, and Kumar (2010).

Key Features

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Encoded plaintext must be loaded into square matrices having the same size as the keys.
- Multiple iterations.

SVK Hill Cipher

Sastry, Varanasi, and Kumar (2010).

Key Features

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Encoded plaintext must be loaded into square matrices having the same size as the keys.
- Multiple iterations.

SVK Hill Cipher

Sastry, Varanasi, and Kumar (2010).

Key Features

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Encoded plaintext must be loaded into square matrices having the same size as the keys.
- Multiple iterations.

Encryption Example

- Suppose Alice wants to send the message "Abbot=A. Square" to Bob.
- Plaintext is converted to decimal numbers between 0 and 255 using the EBCDIC code.
- We use a block size of 4.
- Encoded plaintext is placed into a 4×4 matrix.

$$P = \begin{bmatrix} 193 & 130 & 130 & 150 \\ 163 & 163 & 126 & 193 \\ 175 & 64 & 226 & 152 \\ 164 & 129 & 153 & 133 \end{bmatrix}$$

Encryption Example

- Suppose Alice wants to send the message “Abbot=A. Square” to Bob.
- Plaintext is converted to decimal numbers between 0 and 255 using the EBCDIC code.
- We use a block size of 4.
- Encoded plaintext is placed into a 4×4 matrix.

$$P = \begin{bmatrix} 193 & 130 & 130 & 150 \\ 163 & 163 & 126 & 193 \\ 175 & 64 & 226 & 152 \\ 164 & 129 & 153 & 133 \end{bmatrix}$$

Encryption Example

- Suppose Alice wants to send the message "Abbot=A. Square" to Bob.
- Plaintext is converted to decimal numbers between 0 and 255 using the EBCDIC code.
- We use a block size of 4.
- Encoded plaintext is placed into a 4×4 matrix.

$$P = \begin{bmatrix} 193 & 130 & 130 & 150 \\ 163 & 163 & 126 & 193 \\ 175 & 64 & 226 & 152 \\ 164 & 129 & 153 & 133 \end{bmatrix}$$

Encryption Example

- Suppose Alice wants to send the message “Abbot=A. Square” to Bob.
- Plaintext is converted to decimal numbers between 0 and 255 using the EBCDIC code.
- We use a block size of 4.
- Encoded plaintext is placed into a 4×4 matrix.

$$P = \begin{bmatrix} 193 & 130 & 130 & 150 \\ 163 & 163 & 126 & 193 \\ 175 & 64 & 226 & 152 \\ 164 & 129 & 153 & 133 \end{bmatrix}$$

Encryption Example

Since our block size is 4, Alice and Bob agree on two 4×4 key matrices, K and L.

$$K = \begin{bmatrix} 18 & 33 & 109 & 210 \\ 78 & 43 & 102 & 64 \\ 133 & 17 & 29 & 89 \\ 99 & 87 & 114 & 12 \end{bmatrix} \qquad L = \begin{bmatrix} 19 & 74 & 20 & 103 \\ 88 & 30 & 41 & 19 \\ 211 & 201 & 136 & 87 \\ 77 & 40 & 92 & 126 \end{bmatrix}$$

Alice encrypts P by finding the product

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \pmod{256}$$

Encryption Example

Since our block size is 4, Alice and Bob agree on two 4×4 key matrices, K and L.

$$K = \begin{bmatrix} 18 & 33 & 109 & 210 \\ 78 & 43 & 102 & 64 \\ 133 & 17 & 29 & 89 \\ 99 & 87 & 114 & 12 \end{bmatrix} \qquad L = \begin{bmatrix} 19 & 74 & 20 & 103 \\ 88 & 30 & 41 & 19 \\ 211 & 201 & 136 & 87 \\ 77 & 40 & 92 & 126 \end{bmatrix}$$

Alice encrypts P by finding the product

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \pmod{256}$$

Encryption Example
Permutation: Stage 1

Alice converts the 16 decimal entries in *KPL* to binary, entering them one in each row of a 16×8 matrix.

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \longrightarrow \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 0&1&0&1&0&1&0&0 \\ 0&0&1&1&0&1&0&1 \\ 1&1&0&0&1&1&1&1 \\ 1&1&1&1&1&1&0&0 \\ 1&1&1&1&1&0&1&0 \\ 1&1&1&0&1&1&0&1 \\ 1&0&0&0&0&1&1&0 \\ 0&1&1&1&1&0&1&0 \\ 1&0&0&1&0&1&0&1 \\ 0&1&0&1&1&0&1&0 \\ 0&1&0&1&1&0&0&0 \\ 0&1&1&1&0&0&1&1 \\ 0&1&0&1&1&1&1&0 \\ 1&1&0&1&0&0&1&1 \\ 0&0&1&0&1&1&0&1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 1

Alice converts the 16 decimal entries in *KPL* to binary, entering them one in each row of a 16×8 matrix.

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \longrightarrow \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 0&1&0&1&0&1&0&0 \\ 0&0&1&1&0&1&0&1 \\ 1&1&0&0&1&1&1&1 \\ 1&1&1&1&1&1&0&0 \\ 1&1&1&1&1&0&1&0 \\ 1&1&1&0&1&1&0&1 \\ 1&0&0&0&0&1&1&0 \\ 0&1&1&1&1&0&1&0 \\ 1&0&0&1&0&1&0&1 \\ 0&1&0&1&1&0&1&0 \\ 0&1&0&1&1&0&0&0 \\ 0&1&1&1&0&0&1&1 \\ 0&1&0&1&1&1&1&0 \\ 1&1&0&1&0&0&1&1 \\ 0&0&1&0&1&1&0&1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 1

Alice converts the 16 decimal entries in *KPL* to binary, entering them one in each row of a 16 $\times$ 8 matrix.

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \longrightarrow \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 0&1&0&1&0&1&0&0 \\ 0&0&1&1&0&1&0&1 \\ 1&1&0&0&1&1&1&1 \\ 1&1&1&1&1&1&0&0 \\ 1&1&1&1&1&0&1&0 \\ 1&1&1&0&1&1&0&1 \\ 1&0&0&0&0&1&1&0 \\ 0&1&1&1&1&0&1&0 \\ 1&0&0&1&0&1&0&1 \\ 0&1&0&1&1&0&1&0 \\ 0&1&0&1&1&0&0&0 \\ 0&1&1&1&0&0&1&1 \\ 0&1&0&1&1&1&1&0 \\ 1&1&0&1&0&0&1&1 \\ 0&0&1&0&1&1&0&1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 1

Alice converts the 16 decimal entries in *KPL* to binary, entering them one in each row of a 16×8 matrix.

$$KPL = \begin{bmatrix} 37 & 84 & 53 & 207 \\ 252 & 250 & 237 & 134 \\ 122 & 149 & 90 & 88 \\ 115 & 94 & 211 & 45 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 2

Alice then permutes the binary matrix.

Before permutation:

$$\begin{bmatrix} 0&0&1&0&0&1&0&1\\ 0&1&0&1&0&1&0&0\\ 0&0&1&1&0&1&0&1\\ 1&1&0&0&1&1&1&1\\ 1&1&1&1&1&1&0&0\\ 1&1&1&1&1&0&1&0\\ 1&1&1&0&1&1&0&1\\ 1&0&0&0&0&1&1&0\\ 0&1&1&1&1&0&1&0\\ 1&0&0&1&0&1&0&1\\ 0&1&0&1&1&0&1&0\\ 0&1&0&1&1&0&0&0\\ 0&1&1&1&0&0&1&1\\ 0&1&0&1&1&1&1&0\\ 1&1&0&1&0&0&1&1\\ 0&0&1&0&1&1&0&1 \end{bmatrix}$$

After permutation:

$$\begin{bmatrix} 0&0&1&0&0&0&0&1\\ 1&1&1&0&1&1&1&1\\ 1&0&1&1&1&0&1&0\\ 0&1&1&0&1&1&1&1\\ 0&1&0&1&1&1&0&0\\ 0&1&0&0&0&0&0&0\\ 0&1&1&1&1&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&0&1&1&1&0&0&0\\ 0&0&0&0&0&1&1&0\\ 1&1&1&1&0&0&0&1\\ 1&1&0&0&1&1&1&0\\ 0&1&1&0&0&1&1&1\\ 1&1&1&1&1&0&1&1\\ 1&1&0&0&0&0&1&0\\ 1&1&0&0&1&0&0&1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 2

Alice then permutes the binary matrix.

Before permutation:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

After permutation:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Encryption Example

Permutation: Stage 3

The permuted matrix is then converted back to decimal

$$\begin{bmatrix} 33 & 239 & 186 & 111 \\ 92 & 64 & 127 & 227 \\ 184 & 6 & 241 & 206 \\ 103 & 251 & 194 & 201 \end{bmatrix}$$

This is the final result of one iteration of the SVK Hill Cipher.

Encryption Example

Permutation: Stage 3

The permuted matrix is then converted back to decimal

$$\begin{bmatrix} 33 & 239 & 186 & 111 \\ 92 & 64 & 127 & 227 \\ 184 & 6 & 241 & 206 \\ 103 & 251 & 194 & 201 \end{bmatrix}$$

This is the final result of one iteration of the SVK Hill Cipher.

Encryption Example

Permutation: Stage 3

The permuted matrix is then converted back to decimal

$$\begin{bmatrix} 33 & 239 & 186 & 111 \\ 92 & 64 & 127 & 227 \\ 184 & 6 & 241 & 206 \\ 103 & 251 & 194 & 201 \end{bmatrix}$$

This is the final result of one iteration of the SVK Hill Cipher.

Encryption Example

The entire process is repeated an agreed upon number of times. Along with the key matrices, Alice and Bob also share a number r which dictates how many iterations of the encryption will occur.

In our case, $r = 19$, resulting in the ciphertext

$$C = \begin{bmatrix} 248 & 139 & 132 & 44 \\ 232 & 218 & 237 & 165 \\ 81 & 189 & 155 & 86 \\ 182 & 10 & 65 & 19 \end{bmatrix}$$

Matrix C is then sent over a public channel to Bob.

Encryption Example

The entire process is repeated an agreed upon number of times. Along with the key matrices, Alice and Bob also share a number r which dictates how many iterations of the encryption will occur. In our case, $r = 19$, resulting in the ciphertext

$$C = \begin{bmatrix} 248 & 139 & 132 & 44 \\ 232 & 218 & 237 & 165 \\ 81 & 189 & 155 & 86 \\ 182 & 10 & 65 & 19 \end{bmatrix}$$

Matrix C is then sent over a public channel to Bob.

Encryption Example

The entire process is repeated an agreed upon number of times. Along with the key matrices, Alice and Bob also share a number r which dictates how many iterations of the encryption will occur. In our case, $r = 19$, resulting in the ciphertext

$$C = \begin{bmatrix} 248 & 139 & 132 & 44 \\ 232 & 218 & 237 & 165 \\ 81 & 189 & 155 & 86 \\ 182 & 10 & 65 & 19 \end{bmatrix}$$

Matrix C is then sent over a public channel to Bob.

Encryption Example

The entire process is repeated an agreed upon number of times. Along with the key matrices, Alice and Bob also share a number r which dictates how many iterations of the encryption will occur. In our case, $r = 19$, resulting in the ciphertext

$$C = \begin{bmatrix} 248 & 139 & 132 & 44 \\ 232 & 218 & 237 & 165 \\ 81 & 189 & 155 & 86 \\ 182 & 10 & 65 & 19 \end{bmatrix}$$

Matrix C is then sent over a public channel to Bob.

Decryption Example

- When Bob receives the ciphertext, he knows K, L, and r.
- Bob reverses the permutation (three stages)
- He computes the product

$$K^{-1}CL^{-1} \pmod{256}$$

- He repeats this process r times to obtain the plaintext P.

Decryption Example

- When Bob receives the ciphertext, he knows K, L, and r.
- Bob reverses the permutation (three stages)
- He computes the product

$$K^{-1}CL^{-1} \pmod{256}$$

- He repeats this process r times to obtain the plaintext P.

Decryption Example

- When Bob receives the ciphertext, he knows K, L, and r.
- Bob reverses the permutation (three stages)
- He computes the product

$$K^{-1}CL^{-1} \pmod{256}$$

- He repeats this process r times to obtain the plaintext P.

Decryption Example

- When Bob receives the ciphertext, he knows K, L, and r.
- Bob reverses the permutation (three stages)
- He computes the product

$$K^{-1}CL^{-1} \pmod{256}$$

- He repeats this process r times to obtain the plaintext P.

Avalanche Effect

Suppose we change Alice's original message from "Abbott=A. Square" to "Abbott=A. Sqvare". Here is the binary representation of the final version of the ciphertext after 19 iterations of the encryption process:

Using the modified plaintext:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Using the original plaintext:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}$$

Avalanche Effect

Suppose we change Alice's original message from "Abbott=A. Square" to "Abbott=A. Sqvare". Here is the binary representation of the final version of the ciphertext after 19 iterations of the encryption process:

Using the modified plaintext:

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Using the original plaintext:

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}
$$

Avalanche Effect

Suppose we change Alice's original message from "Abbott=A. Square" to "Abbott=A. Sqvare". Here is the binary representation of the final version of the ciphertext after 19 iterations of the encryption process:

Using the modified plaintext:

$$\begin{bmatrix} 1&1&1&1&0&0&0&1\\ 0&0&1&0&0&1&0&1\\ 1&1&1&1&1&1&1&0\\ 1&0&1&0&0&0&1&1\\ 0&0&1&0&1&0&0&1\\ 0&0&1&0&0&0&0&1\\ 0&0&1&1&1&0&0&0\\ 0&0&0&1&0&1&1&0\\ 0&0&1&1&1&0&1&0\\ 1&0&1&1&0&1&1&1\\ 1&0&0&0&0&0&1&0\\ 1&0&1&0&1&0&1&0\\ 0&1&1&1&0&1&1&0\\ 0&0&1&0&1&1&1&0\\ 0&0&0&0&1&1&1&1\\ 1&0&0&1&0&0&0&0 \end{bmatrix}$$

Using the original plaintext:

$$\begin{bmatrix} 1&1&1&1&1&0&0&0\\ 1&0&0&0&1&0&1&1\\ 1&0&0&0&0&1&0&0\\ 0&0&1&0&1&1&0&0\\ 1&1&1&0&1&0&0&0\\ 1&1&0&1&1&0&1&0\\ 1&1&1&0&1&1&0&1\\ 1&0&1&0&0&1&0&1\\ 0&1&0&1&0&0&0&1\\ 1&0&1&1&1&1&0&1\\ 1&0&0&1&1&0&1&1\\ 0&1&0&1&0&1&1&0\\ 1&0&1&1&0&1&1&0\\ 0&0&0&0&1&0&1&0\\ 0&1&0&0&0&0&0&1\\ 0&0&0&1&0&0&1&1 \end{bmatrix}$$

Avalanche Effect

Similarly, we can observe the avalanche effect on a slight change to one of the key matrices. Suppose we change the first row, first column entry of the key matrix K from 18 to 17. After applying the same encryption scheme for 19 iterations:

Using the modified key:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Using the original key:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Avalanche Effect

Similarly, we can observe the avalanche effect on a slight change to one of the key matrices. Suppose we change the first row, first column entry of the key matrix K from 18 to 17. After applying the same encryption scheme for 19 iterations:

Using the modified key:

$$\begin{bmatrix} 1&0&1&0&0&0&1&0\\ 1&0&1&0&0&0&1&0\\ 0&1&1&1&1&1&0&1\\ 0&0&1&0&0&0&1&1\\ 1&0&1&1&0&1&1&1\\ 1&1&0&1&1&1&0&0\\ 0&1&1&0&1&1&0&0\\ 1&1&0&1&1&1&1&0\\ 0&0&1&1&1&0&0&1\\ 1&0&0&1&0&0&1&1\\ 0&1&1&1&0&1&1&0\\ 1&1&1&1&0&1&0&1\\ 1&0&0&1&1&1&1&0\\ 1&0&1&0&0&0&0&0\\ 0&0&0&0&1&1&1&1\\ 1&0&0&0&1&1&1&1 \end{bmatrix}$$

Using the original key:

$$\begin{bmatrix} 1&1&1&1&1&0&0&0\\ 1&0&0&0&1&0&1&1\\ 1&0&0&0&0&1&0&0\\ 0&0&1&0&1&1&0&0\\ 1&1&1&0&1&0&0&0\\ 1&1&0&1&1&0&1&0\\ 1&1&1&0&1&1&0&1\\ 1&0&1&0&0&1&0&1\\ 0&1&0&1&0&0&0&1\\ 1&0&1&1&1&1&0&1\\ 1&0&0&1&1&0&1&1\\ 0&1&0&1&0&1&1&0\\ 1&0&1&1&0&1&1&0\\ 0&0&0&0&1&0&1&0\\ 0&1&0&0&0&0&0&1\\ 0&0&0&1&0&0&1&1 \end{bmatrix}$$

Avalanche Effect

Similarly, we can observe the avalanche effect on a slight change to one of the key matrices. Suppose we change the first row, first column entry of the key matrix K from 18 to 17. After applying the same encryption scheme for 19 iterations:

Using the modified key:

$$\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}$$

Using the original key:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}$$

Summary of the SVK Hill Cipher

Summary

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Multiple iterations.
- Secure against known-plaintext attack.

Summary of the SVK Hill Cipher

Summary

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Multiple iterations.
- Secure against known-plaintext attack.

Summary of the SVK Hill Cipher

Summary

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Multiple iterations.
- Secure against known-plaintext attack.

Summary of the SVK Hill Cipher

Summary

- Uses a pair of key matrices (multiplied on the left and right) and a permutation scheme.
- Multiple iterations.
- Secure against known-plaintext attack.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

TF Hill Cipher

Toorani and Falahati (2009).

Key Features

- Variant of the *Affine Hill Cipher* $Y = XK + V \pmod{m}$.
- In TFHC, a new vector V is used for each plaintext block.

Encryption

- Encryption rule $Y_t = v_0 X_t K + V_t \pmod{p}$
- We use a unique v_0 and V_t for each plaintext block X_t.
- For each plaintext block, a new a_t is recursively generated from a random initial value a_0 using a one-way hash function.
- Then the constant v_0 and vector V_t are generated using a_t and the key matrix K.

Encryption example

Suppose our plaintext is "over the hill and through the woods". We might encode it using the following alphabet with $p = 29$:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

[space]	' [apostrophe]	. [period]	
26	27	28	

Our encoded plaintext would be

[14 21 4 17 26 19 7 4 26 7 8 11 11 26 0 13 3 26 19 7 17 14 20 6 7 26 19 7 4 26 22 14 14 3 18]

Encryption example

Suppose our plaintext is "over the hill and through the woods". We might encode it using the following alphabet with $p = 29$:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

[space]	' [apostrophe]	. [period]	
26	27	28	

Our encoded plaintext would be

[14 21 4 17 26 19 7 4 26 7 8 11 11 26 0 13 3 26 19 7 17 14 20 6 7 26 19 7 4 26 22 14 14 3 18]

Encryption example

Suppose our plaintext is "over the hill and through the woods". We might encode it using the following alphabet with $p = 29$:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

[space]	' [apostrophe]	. [period]	
26	27	28	

Our encoded plaintext would be

[14 21 4 17 26 19 7 4 26 7 8 11 11 26 0 13 3 26 19 7 17 14 20 6 7 26 19 7 4 26 22 14 14 3 18]

Encryption example

Suppose we choose a block size of $n = 5$.

Let the key matrix be

$$K = \begin{bmatrix} 1 & 5 & 17 & 25 & 16 \\ 9 & 0 & 18 & 19 & 23 \\ 26 & 4 & 10 & 13 & 2 \\ 3 & 4 & 8 & 9 & 22 \\ 7 & 24 & 15 & 17 & 12 \end{bmatrix}$$

Encryption example

Suppose we choose a block size of $n = 5$.

Let the key matrix be

$$K = \begin{bmatrix} 1 & 5 & 17 & 25 & 16 \\ 9 & 0 & 18 & 19 & 23 \\ 26 & 4 & 10 & 13 & 2 \\ 3 & 4 & 8 & 9 & 22 \\ 7 & 24 & 15 & 17 & 12 \end{bmatrix}$$

Encryption example

Step 1

We choose a random a_0 and b such that $0 < a_0 < p - 1$ and $1 < b < n^2$. We choose

$$a_0 = 20$$

$$b = 17$$

since $0 < a_0 < 28$, and since $1 < b < 25$.

Step 2

Next we compute $r = a_0 k_{ij} \pmod{p}$ where $i = \lceil b/n \rceil$ and $j = b - n(i - 1)$. We find

$$r = 22$$

Encryption example

Step 1

We choose a random a_0 and b such that $0 < a_0 < p - 1$ and $1 < b < n^2$. We choose

$$a_0 = 20$$

$$b = 17$$

since $0 < a_0 < 28$, and since $1 < b < 25$.

Step 2

Next we compute $r = a_0 k_{ij} \pmod{p}$ where $i = \lceil b/n \rceil$ and $j = b - n(i - 1)$. We find

$$r = 22$$

Encryption example

Step 1

We choose a random a_0 and b such that $0 < a_0 < p - 1$ and $1 < b < n^2$. We choose

$$a_0 = 20$$

$$b = 17$$

since $0 < a_0 < 28$, and since $1 < b < 25$.

Step 2

Next we compute $r = a_0 k_{ij} \pmod{p}$ where $i = \lceil b/n \rceil$ and $j = b - n(i-1)$. We find

$$r = 22$$

Encryption example

Step 3

We encode the plaintext into row vectors X_t of length $n = 5$

$$X_1 = [14\ 21\ 4\ 17\ 26]$$
$$X_2 = [19\ 7\ 4\ 26\ 7]$$
$$X_3 = [8\ 11\ 11\ 26\ 0]$$
$$X_4 = [13\ 3\ 26\ 19\ 7]$$
$$X_5 = [17\ 14\ 20\ 6\ 7]$$
$$X_6 = [26\ 19\ 7\ 4\ 26]$$
$$X_7 = [22\ 14\ 14\ 3\ 18]$$

Encryption example

Step 3

We encode the plaintext into row vectors X_t of length $n = 5$

$$
\begin{aligned}
X_1 &= [14\ 21\ 4\ 17\ 26] \\
X_2 &= [19\ 7\ 4\ 26\ 7] \\
X_3 &= [8\ 11\ 11\ 26\ 0] \\
X_4 &= [13\ 3\ 26\ 19\ 7] \\
X_5 &= [17\ 14\ 20\ 6\ 7] \\
X_6 &= [26\ 19\ 7\ 4\ 26] \\
X_7 &= [22\ 14\ 14\ 3\ 18]
\end{aligned}
$$

Encryption example

Step 4

For each plaintext block X_t, we compute $a_t = H(a_{t-1})$ using a recursive one-way hash function H. In our case, we will use the MD5 hash function. For example, to encrypt the first plaintext block X_1, we calculate $a_1 = H(a_0)$, obtaining

$$a_1 = H(20) = 203295115078782523880027993804846545796$$

Step 5

We then compute v_0: If a_t is invertible mod p, that is $a_t \not\equiv 0 \pmod{p}$, we let $v_0 = a_t \pmod{p}$. Otherwise $v_0 = 1$. In our case

$$v_0 = a_1 \pmod{29} = 7$$

Encryption example

Step 4

For each plaintext block X_t, we compute $a_t = H(a_{t-1})$ using a recursive one-way hash function H. In our case, we will use the MD5 hash function. For example, to encrypt the first plaintext block X_1, we calculate $a_1 = H(a_0)$, obtaining

$$a_1 = H(20) = 203295115078782523880027993804846545796$$

Step 5

We then compute v_0: If a_t is invertible mod p, that is $a_t \neq 0 \pmod{p}$, we let $v_0 = a_t \pmod{p}$. Otherwise $v_0 = 1$. In our case

$$v_0 = a_1 \pmod{29} = 7$$

Encryption example

Step 4

For each plaintext block X_t, we compute $a_t = H(a_{t-1})$ using a recursive one-way hash function H. In our case, we will use the MD5 hash function. For example, to encrypt the first plaintext block X_1, we calculate $a_1 = H(a_0)$, obtaining

$$a_1 = H(20) = 203295115078782523880027993804846545796$$

Step 5

We then compute v_0: If a_t is invertible mod p, that is $a_t \not\equiv 0 \pmod{p}$, we let $v_0 = a_t \pmod{p}$. Otherwise $v_0 = 1$. In our case

$$v_0 = a_1 \pmod{29} = 7$$

Encryption example

Step 6

We then compute row vector $V_t = [v_1 \; v_2 \; \cdots \; v_n]$ with the recursive expression $v_i = k_{ij} + \tilde{v}_{i-1} a_t \pmod p$ for $i = 1, \ldots, n$ and $j = (v_{i-1} \mod n) + 1$, in which $\tilde{v}_{i-1} = 2^{\lceil \gamma/2 \rceil} + (v_{i-1} \mod 2^{\lceil \gamma/2 \rceil})$ and $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ denotes the bit length of v_{i-1}. In our case,

$$\begin{array}{lllll}
i = 1 & j = 3 & \gamma = 3 & \tilde{v}_0 = 7 & v_1 = 8 \\
i = 2 & j = 4 & \gamma = 4 & \tilde{v}_1 = 4 & v_2 = 18 \\
i = 3 & j = 4 & \gamma = 5 & \tilde{v}_2 = 10 & v_3 = 25 \\
i = 4 & j = 1 & \gamma = 5 & \tilde{v}_3 = 9 & v_4 = 8 \\
i = 5 & j = 4 & \gamma = 4 & \tilde{v}_4 = 4 & v_5 = 16
\end{array}$$

So $V_1 = [8 \; 18 \; 25 \; 8 \; 16]$.

Encryption example

Step 6

We then compute row vector $V_t = [v_1 \ v_2 \ \cdots \ v_n]$ with the recursive expression $v_i = k_{ij} + \tilde{v}_{i-1} a_t \pmod p$ for $i = 1, \ldots, n$ and $j = (v_{i-1} \mod n) + 1$, in which $\tilde{v}_{i-1} = 2^{\lceil \gamma/2 \rceil} + \left(v_{i-1} \mod 2^{\lceil \gamma/2 \rceil}\right)$ and $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ denotes the bit length of v_{i-1}. In our case,

$i = 1$	$j = 3$	$\gamma = 3$	$\tilde{v}_0 = 7$	$v_1 = 8$
$i = 2$	$j = 4$	$\gamma = 4$	$\tilde{v}_1 = 4$	$v_2 = 18$
$i = 3$	$j = 4$	$\gamma = 5$	$\tilde{v}_2 = 10$	$v_3 = 25$
$i = 4$	$j = 1$	$\gamma = 5$	$\tilde{v}_3 = 9$	$v_4 = 8$
$i = 5$	$j = 4$	$\gamma = 4$	$\tilde{v}_4 = 4$	$v_5 = 16$

So $V_1 = [8 \ 18 \ 25 \ 8 \ 16]$.

Encryption example

Step 6

We then compute row vector $V_t = [v_1 \ v_2 \ \cdots \ v_n]$ with the recursive expression $v_i = k_{ij} + \tilde{v}_{i-1} a_t \pmod p$ for $i = 1, \ldots, n$ and $j = (v_{i-1} \mod n) + 1$, in which $\tilde{v}_{i-1} = 2^{\lceil \gamma/2 \rceil} + \left(v_{i-1} \mod 2^{\lceil \gamma/2 \rceil}\right)$ and $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ denotes the bit length of v_{i-1}. In our case,

$$\begin{array}{lllll} i=1 & j=3 & \gamma=3 & \tilde{v}_0=7 & v_1=8 \\ i=2 & j=4 & \gamma=4 & \tilde{v}_1=4 & v_2=18 \\ i=3 & j=4 & \gamma=5 & \tilde{v}_2=10 & v_3=25 \\ i=4 & j=1 & \gamma=5 & \tilde{v}_3=9 & v_4=8 \\ i=5 & j=4 & \gamma=4 & \tilde{v}_4=4 & v_5=16 \end{array}$$

So $V_1 = [8\ 18\ 25\ 8\ 16]$.

Encryption example

Step 6

We then compute row vector $V_t = [v_1 \; v_2 \; \cdots \; v_n]$ with the recursive expression $v_i = k_{ij} + \tilde{v}_{i-1} a_t \pmod{p}$ for $i = 1, \ldots, n$ and $j = (v_{i-1} \mod n) + 1$, in which $\tilde{v}_{i-1} = 2^{\lceil \gamma/2 \rceil} + (v_{i-1} \mod 2^{\lceil \gamma/2 \rceil})$ and $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ denotes the bit length of v_{i-1}. In our case,

$$
\begin{array}{lllll}
i = 1 & j = 3 & \gamma = 3 & \tilde{v}_0 = 7 & v_1 = 8 \\
i = 2 & j = 4 & \gamma = 4 & \tilde{v}_1 = 4 & v_2 = 18 \\
i = 3 & j = 4 & \gamma = 5 & \tilde{v}_2 = 10 & v_3 = 25 \\
i = 4 & j = 1 & \gamma = 5 & \tilde{v}_3 = 9 & v_4 = 8 \\
i = 5 & j = 4 & \gamma = 4 & \tilde{v}_4 = 4 & v_5 = 16
\end{array}
$$

So $V_1 = [8 \; 18 \; 25 \; 8 \; 16]$.

Encryption example

Step 7

We encrypt each X_t, as $Y_t = v_0 X_t K + V_t \pmod{p}$. In the case of X_1, we have

$$Y_1 = (7)\,[14\ 21\ 4\ 17\ 26]\,K + V_1 \pmod{p} = [18\ 12\ 5\ 7\ 21]$$

Encryption example

Step 7

We encrypt each X_t, as $Y_t = v_0 X_t K + V_t \pmod{p}$. In the case of X_1, we have

$$Y_1 = (7)\,[14\ 21\ 4\ 17\ 26]\,K + V_1 \pmod{p} = [18\ 12\ 5\ 7\ 21]$$

Encryption example

Step 8

We repeat this for all the other plaintext blocks, yielding:

$$
\begin{aligned}
Y_1 &= [18\ 12\ 5\ 7\ 21] \\
Y_2 &= [18\ 22\ 14\ 5\ 21] \\
Y_3 &= [23\ 24\ 13\ 14\ 8] \\
Y_4 &= [20\ 9\ 1\ 2\ 4] \\
Y_5 &= [10\ 18\ 26\ 1\ 26] \\
Y_6 &= [5\ 18\ 8\ 3\ 24] \\
Y_7 &= [4\ 5\ 0\ 12\ 0]
\end{aligned}
$$

We pass these ciphertext blocks along with $b = 17$ and $r = 22$ over a public channel to the recipient.

Encryption example

Step 8

We repeat this for all the other plaintext blocks, yielding:

$$
\begin{aligned}
Y_1 &= [18\ 12\ 5\ 7\ 21] \\
Y_2 &= [18\ 22\ 14\ 5\ 21] \\
Y_3 &= [23\ 24\ 13\ 14\ 8] \\
Y_4 &= [20\ 9\ 1\ 2\ 4] \\
Y_5 &= [10\ 18\ 26\ 1\ 26] \\
Y_6 &= [5\ 18\ 8\ 3\ 24] \\
Y_7 &= [4\ 5\ 0\ 12\ 0]
\end{aligned}
$$

We pass these ciphertext blocks along with $b = 17$ and $r = 22$ over a public channel to the recipient.

Encryption example

Step 8

We repeat this for all the other plaintext blocks, yielding:

$$
\begin{aligned}
Y_1 &= [18\ 12\ 5\ 7\ 21] \\
Y_2 &= [18\ 22\ 14\ 5\ 21] \\
Y_3 &= [23\ 24\ 13\ 14\ 8] \\
Y_4 &= [20\ 9\ 1\ 2\ 4] \\
Y_5 &= [10\ 18\ 26\ 1\ 26] \\
Y_6 &= [5\ 18\ 8\ 3\ 24] \\
Y_7 &= [4\ 5\ 0\ 12\ 0]
\end{aligned}
$$

We pass these ciphertext blocks along with $b = 17$ and $r = 22$ over a public channel to the recipient.

Encryption example

Step 8

We repeat this for all the other plaintext blocks, yielding:

$$
\begin{aligned}
Y_1 &= [18\ 12\ 5\ 7\ 21] \\
Y_2 &= [18\ 22\ 14\ 5\ 21] \\
Y_3 &= [23\ 24\ 13\ 14\ 8] \\
Y_4 &= [20\ 9\ 1\ 2\ 4] \\
Y_5 &= [10\ 18\ 26\ 1\ 26] \\
Y_6 &= [5\ 18\ 8\ 3\ 24] \\
Y_7 &= [4\ 5\ 0\ 12\ 0]
\end{aligned}
$$

We pass these ciphertext blocks along with $b = 17$ and $r = 22$ over a public channel to the recipient.

Decryption example

1. Bob knows the secret key K. He receives the ciphertext blocks Y_t sent by Alice, $b = 17$, and $r = 22$ over public channel.
2. He computes $u = k_{ij}^{-1} \pmod{p}$ and $a_0 = ru \pmod{p}$ in which $i = \lceil b/n \rceil$ and $j = b - n(i-1)$ and finds

$$a_0 = 20$$

3. Since he now has $a_0 = 20$, he computes V_t for each ciphertext block just as Alice did.
4. He then decrypts each ciphertext block by $X_t = v_0^{-1}(Y_t - V_t)K^{-1} \pmod{p}$, obtaining all the plaintext matrices.

Decryption example

1. Bob knows the secret key K. He receives the ciphertext blocks Y_t sent by Alice, $b = 17$, and $r = 22$ over public channel.
2. He computes $u = k_{ij}^{-1} \pmod{p}$ and $a_0 = ru \pmod{p}$ in which $i = \lceil b/n \rceil$ and $j = b - n(i-1)$ and finds

$$a_0 = 20$$

3. Since he now has $a_0 = 20$, he computes V_t for each ciphertext block just as Alice did.
4. He then decrypts each ciphertext block by $X_t = v_0^{-1}(Y_t - V_t)K^{-1} \pmod{p}$, obtaining all the plaintext matrices.

Decryption example

1. Bob knows the secret key K. He receives the ciphertext blocks Y_t sent by Alice, $b = 17$, and $r = 22$ over public channel.
2. He computes $u = k_{ij}^{-1} \pmod{p}$ and $a_0 = ru \pmod{p}$ in which $i = \lceil b/n \rceil$ and $j = b - n(i-1)$ and finds

$$a_0 = 20$$

3. Since he now has $a_0 = 20$, he computes V_t for each ciphertext block just as Alice did.
4. He then decrypts each ciphertext block by $X_t = v_0^{-1}(Y_t - V_t)K^{-1} \pmod{p}$, obtaining all the plaintext matrices.

Decryption example

1. Bob knows the secret key K. He receives the ciphertext blocks Y_t sent by Alice, $b = 17$, and $r = 22$ over public channel.
2. He computes $u = k_{ij}^{-1} \pmod p$ and $a_0 = ru \pmod p$ in which $i = \lceil b/n \rceil$ and $j = b - n(i-1)$ and finds
$$a_0 = 20$$
3. Since he now has $a_0 = 20$, he computes V_t for each ciphertext block just as Alice did.
4. He then decrypts each ciphertext block by $X_t = v_0^{-1}\,(Y_t - V_t)\,K^{-1} \pmod p$, obtaining all the plaintext matrices.

Summary of Hill Cipher

Summary

- Variant on the Affine Hill Cipher.
- A new vector V is generated by a secure hash for each block of plaintext.
- Secure against the known-plaintext attack.

Summary of Hill Cipher

Summary

- Variant on the Affine Hill Cipher.
- A new vector V is generated by a secure hash for each block of plaintext.
- Secure against the known-plaintext attack.

Summary of Hill Cipher

Summary

- Variant on the Affine Hill Cipher.
- A new vector V is generated by a secure hash for each block of plaintext.
- Secure against the known-plaintext attack.

Summary of Hill Cipher

Summary

- Variant on the Affine Hill Cipher.
- A new vector V is generated by a secure hash for each block of plaintext.
- Secure against the known-plaintext attack.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.

Summary

- The classic Hill Cipher is insecure, subject to known-plaintext attack.
- We have presented two secure variants on Hill's cipher:
  - Sastry, Varanasi, and Kumar (2010)
  - Toorani and Falahati (2009)
- In both cases, the Hill Cipher has been made secure against the known-plaintext attack.
- These two strong symmetric ciphers are ready for use in the real world.