NTNU

Norwegian University of Science and Technology

Department of Information Security and Communication Technology

TTM4135
Applied Cryptography and Network Security

Classical Cipher Cryptanalysis

Report

Author:
Orlando Tomeo

Spring 2026

1 Methodology and Statistical Overview

The alphabet used for this analysis consists of the standard 26 uppercase English letters plus three punctuation marks (`,` `.` `-`), totaling 29 characters ($N = 29$).

Due to this extended alphabet, standard online cryptanalysis tools (which typically default to the standard 26-letter alphabet) were found to be unsuitable. Consequently, I developed custom Python scripts to handle the specific modulo-29 arithmetic and character mapping for all statistical analyses and decryption tasks.

The first step in analyzing the provided ciphertexts (labeled 0, 1, 2, and 3) was to compute the frequency distribution of single characters (1-grams), digrams, and trigrams, along with the Index of Coincidence (IC).

The IC is defined as:

$$IC = \frac{\sum_{i=1}^{c} f_i(f_i - 1)}{N(N-1)} \tag{1}$$

For the English language, the expected IC is approximately 0.066. For random text (uniform distribution), the expected IC for an alphabet of size 29 is $1/29 \approx 0.034$.

Table 1 summarizes the initial metrics for each ciphertext.

Cipher ID	Length	IC	Preliminary Classification
0	1048	0.0622	Monoalphabetic (Substitution)
1	1212	0.0632	Monoalphabetic (Caesar)
2	974	0.0371	Polygraphic (Hill)
3	1067	0.0395	Polyalphabetic (Vigenère)

Table 1: Statistical summary of the ciphertexts.

2 Cipher 0: Monoalphabetic Substitution

2.1 Identification

Ciphertext 0 exhibited an IC of 0.0622, closely matching the expected IC for English (0.066). This strongly indicated a monoalphabetic substitution cipher. Since a check for Caesar shifts failed to produce readable text, I concluded it was a random substitution alphabet.

To perform the statistical analysis, I utilized a custom Python script leveraging the `collections.Counter` library to generate frequency distributions for 1-grams, 2-grams, and 3-grams tailored to the 29-character alphabet.

Table 2 shows the frequency analysis confirming the substitution nature.

1-grams		2-grams		3-grams	
Char	%	Digram	%	Trigram	%
C	12.12	UI	3.53	UIC	1.63
U	8.30	IC	3.44	TKJ	1.34
E	8.11	TK	2.96	IC,	1.15
I	7.54	C,	2.29	VEL	0.67
K	7.16	KJ	1.53	UIT	0.57

Table 2: Top frequencies for Cipher 0. Note the high frequency of 'UIC' corresponding to 'THE'.

2.2 Cryptanalysis (Iterative Reconstruction)

I recovered the key through a four-stage iterative process, utilizing frequency analysis and pattern matching (crib dragging).

2.2.1 Stage 1: High-Frequency Mapping

The most frequent character in the ciphertext is `'C'` (12.12%), followed by `'U'` (8.30%). In the reference statistics provided for this assignment, `'E'` (11.93%) and `'T'` (8.8%) are the dominant letters.

- **Hypothesis 1:** $C \to e$, $U \to t$.

2.2.2 Stage 2: Trigram Analysis ("THE")

I analyzed the most frequent trigram, `'UIC'` (1.63%). Applying the hypothesis from Stage 1 ($U \to t, C \to e$), this trigram becomes `"t?e"`. The only statistically significant fit in English is the word **"THE"** (reference frequency 1.58%).

This hypothesis is further validated by the digram distribution: if $UIC \to THE$, then $UI \to TH$ and $IC \to HE$. Table 2 confirms that UI (3.53%) and IC (3.44%) are indeed the top two digrams, matching the prevalence of TH and HE in English.

- **Deduction:** $I \to h$.

2.2.3 Stage 3: Pattern Matching ("-ING")

With the partial mapping, I searched for common suffixes. The trigram `'TKJ'` (1.34%) appeared frequently at the end of word structures. I hypothesized this corresponded to the common gerund suffix **"-ING"**.

- **Deduction:** $T \to i$, $K \to n$, $J \to g$.

2.2.4 Stage 4: Word Completion

Applying the mappings derived so far revealed partial words, allowing me to deduce the remaining letters by context.

- The pattern `"e.en"` appeared, suggesting the word **"even"**. Thus, the symbol `'.'` maps to `'v'`.
- The ciphertext segment `"VEL"` mapped to the structure `"wAs"` (was), deducing $V \to w, E \to a, L \to s$.

2.2.5 Implementation Details: Evolution of the Plaintext

To verify these hypotheses, I implemented an incremental decryption script in Python. The following output demonstrates how the plaintext evolved from unintelligible text to coherent English at each step:

```
# Incremental Decryption Logic (Conceptual Snippet)
map_step1 = {'C': 'e', 'U': 't'}
map_step2 = {'C': 'e', 'U': 't', 'I': 'h'} # + THE
map_step3 = {'C': 'e', 'U': 't', 'I': 'h', 'T': 'i', 'K': 'n', 'J': 'g'} # + ING
# ... (Continuing for all steps)
```

```
--- Step 1: Hypothesis C=e, U=t ---
e.eKVITQetIeYIESGeeKLEYTKJANZZNKMQEAetITKJLLWLEKIESGeeKANKLA...

--- Step 2: Trigram UIC -> THE (I=h) ---
e.eKVhTQethеYhESGeeKLEYTKJANZZNKMQEAethTKJLLWLEKhESGeeKANKLA...
```

```
--- Step 3: Suffix TKJ -> ING ---
e.enVhiQetheYhESGeenLEYingANZZNnMQEAethingLLWLEnhESGeenANnLA...

--- Step 4: Word Deduction (even, was) ---
evenwhiQetheYhaSGeensaYingANZZNnMQaAethingssWsanhaSGeenANnsA...
```

As seen in Step 4, coherent fragments like `"even whi... they ... been saying ... things"` emerge, confirming the partial key and allowing for the full recovery of the alphabet.

2.3 Result

Following this chain of deductions, the full key was reconstructed. The plaintext was recovered successfully:

```
evenwhiletheyhadbeensayingcommonplacethingssusanhadbeenconsciousoftheexcite
mentofintimacywhichseemednotonlytolaybaresomethinginherwbutinthetreesandthe
skywandtheprogressofhisspeechwhichseemedinevitablewaspositivelypainfultoherw
fornohumanbeinghadevercomesoclosetoherbeforeAshewasstruckmotionlessashisspee
chwentonwandherheartgavegreatseparateleapsatthelastwordsAshesatwithherfinger
scurledroundastonewlookingstraightinfrontofherdownthemountainovertheplainAso
thenwithadactuallyhappenedtoherwaproposalofarthurlookedroundatherhisfacewaso
ddlytwistedAshewasdrawingherbreathwithsuchdifficultythatshecouldhardlyanswer
AyoumighthaveknownAheseizedherinhisarmsagainandagainandagaintheyclaspedeacho
therwmurmuringinarticulatelyAwellwsighedarthurwsinkingbackonthegroundwthatst
hemostwonderfulthingthatseverhappenedtomeAhelookedasifhewereretryingtoputthing
sseeninadreambesiderealthingsAtherewasalongsilenceAitsthemostperfectthingint
heworldwsusanstatedwverygentlyandwithgreatconvictionAitwasnolongermerelyapro
posalofmarriagewbutofmarriagewitharthurwithwhomshewasinloveA
```

3 Cipher 1: Caesar Cipher

3.1 Identification

Ciphertext 1 had the highest IC (0.0632). The frequency distribution appeared identical to standard English but shifted. The character `'X'` was the most frequent (13.78%), followed by `'J'` (8.42%).

3.2 Cryptanalysis

Assuming the most frequent letter `'X'` maps to standard English `'E'`:

$$\begin{aligned} \text{Index('X')} &= 23 \\ \text{Index('E')} &= 4 \\ k &= (23 - 4) \pmod{29} = 19 \end{aligned}$$

3.2.1 Implementation Details

The decryption was automated using a modulo shift operation:

```python
SHIFT = 19
# Decryption loop
plain1 = "".join(ALPHABET[(ALPHABET.index(c) - SHIFT) % 29] for c in cipher1)
```

3.3 Result

Applying a shift of 19 revealed the plaintext:

```
THEUSUALEFFECTOFTAKINGAWAYALLDESIREFORCOMMUNICATIONBYMAKINGTHEIRWORDSSOUNDTHI
NANDSMALLAND,AFTERWALKINGROUNDTHEDECKTHREEORFOURTIMES,THEYCLUSTEREDTOGETHER,Y
AWNINGDEEPLY,ANDLOOKINGATTHESAMESPOTOFDEEPGLOOMONTHEBANKS.MURMURINGVERYLOWINT
HERHYTHMICALTONEOFONEOPPRESSEDBYTHEAIR,MRS.FLUSHINGBEGANTOWONDERWHERETHEYWERE
TOSLEEP,FORTHEYCOULDNOTSLEEPDOWNSTAIRS,THEYCOULDNOTSLEEPINADOGHOLESMELLINGOFO
IL,THEYCOULDNOTSLEEPONDECK,THEYCOULDNOTSLEEP--SHEYAWNEDPROFOUNDLY.ITWASASHELE
NHADFORESEENTHEQUESTIONOFNAKEDNESSHADRISENALREADY,ALTHOUGHTHEYWEREHALFASLEEP,
ANDALMOSTINVISIBLETOEACHOTHER.WITHST.JOHNSHELPSHESTRETCHEDANAWNING,ANDPERSUAD
EDMRS.FLUSHINGTHATSHECOULDTAKEOFFHERCLOTHESBEHINDTHIS,ANDTHATNOONEWOULDNOTICE
IFBYCHANCESOMEPARTOFHERWHICHHADBEENCONCEALEDFORFORTY-FIVEYEARSWASLAIDBARETOTH
EHUMANEYE.MATTRESSESWERETHROWNDOWN,RUGSPROVIDED,ANDTHETHREEWOMENLAYNEAREACHOT
HERINTHESOFTOPENAIR.THEGENTLEMEN,HAVINGSMOKEDACERTAINNUMBEROFCIGARETTES,DROPP
EDTHEGLOWINGENDSINTOTHERIVER,ANDLOOKEDFORATIMEATTHERIPPLESWRINKLINGTHEBLACKWA
TERBENEATHTHEM,UNDRESSEDTOO,ANDLAYDOWNATTHEOTHERENDOFTHEBOAT.THEYWEREVERYTIRE
D,ANDCURTAINEDFROMEACHOTHERBYTHEDARKNESS.THELIGHTFROMONELANTERNFELLUPONAFEWRO
PES,AFEWPLANKSOFTHEDECK,ANDTHERAILOFTHEBOAT,BUTBEYONDTHAT
```

4 Cipher 2: Hill Cipher (2×2)

4.1 Identification

Ciphertext 2 was identified as a linear polygraphic cipher due to its low IC (0.0371) and flat digram distribution, which masks single-letter frequencies but preserves linear relationships.

4.2 Cryptanalysis

Breaking the Hill cipher required recovering the decryption matrix $D = K^{-1}$. I attempted two different attack vectors.

4.2.1 Attempt 1: Context-Based Crib Attack (Failed)

Initially, I hypothesized that the plaintext might begin with standard cryptographic headers often found in such challenges (e.g., "THEHILLCIPHER", "LESTERHILL"). I developed a script to test these "cribs" against the first N characters of the ciphertext. This approach yielded non-invertible matrices or gibberish output, leading me to reject the hypothesis that the text started with a standard header.

4.2.2 Attempt 2: Algebraic Digram Attack (Successful)

I shifted to a ciphertext-only attack based on frequency analysis. I identified the most frequent ciphertext digrams and hypothesized they mapped to common English digrams ("TH", "HE").

- **Hypothesis 1:** Ciphertext **"BO"** (C_1) $\rightarrow$ Plaintext **"HE"** (P_1)
- **Hypothesis 2:** Ciphertext **"XF"** (C_2) $\rightarrow$ Plaintext **"TH"** (P_2)

Using the numerical indices ($A = 0, \ldots, - = 28$), I established the linear system in $\mathbb{Z}_{29}$:

$$P = \begin{pmatrix} 19 & 7 \\ 7 & 4 \end{pmatrix}, \quad C = \begin{pmatrix} 23 & 1 \\ 5 & 14 \end{pmatrix}$$

Note: The columns represent the digrams TH (19, 7) and HE (7, 4) for P, and XF (23, 5) and BO (1, 14) for C.

The decryption matrix $D = K^{-1}$ satisfies $D \cdot C = P \pmod{29}$. I solved for D:

$$D = P \cdot C^{-1} \pmod{29}$$

Validation of the resulting matrix included checking the determinant to ensure invertibility.

$$D = \begin{pmatrix} 15 & 16 \\ 19 & 1 \end{pmatrix}$$

The determinant of the calculated decryption matrix D was -289. Since $\gcd(-289, 29) = 1$, the matrix is a valid, invertible cryptographic key.

By mapping the matrix values back to the alphabet ($15 \to \mathbf{P}$, $16 \to \mathbf{Q}$, $19 \to \mathbf{T}$, $1 \to \mathbf{B}$), the final key characters (read row by row) are:

Key: PQTB

4.2.3 Implementation Details

The following Python code was used to solve the linear algebra system and verify the matrix invertibility:

```python
def solve_hill_linear_algebra():
    # TH -> [19, 7], HE -> [7, 4]
    P = np.array([[19, 7], [7, 4]])
    # XF -> [23, 5], BO -> [1, 14]
    C = np.array([[23, 1], [5, 14]])

    # Solve D = P * C^-1
    det_c = int(np.round(np.linalg.det(C)))
    det_inv = pow(det_c, -1, 29) # Modular inverse

    # Adjugate matrix for C
    C_adj = np.array([[C[1,1], -C[0,1]], [-C[1,0], C[0,0]]])
    C_inv_matrix = (det_inv * C_adj) % 29

    # Calculate Key
    D = np.dot(P, C_inv_matrix) % 29
    return D
```

4.3 Result

The plaintext is fully decrypted below:

TODRIFTPASTEACHOTHERINSILENCE.IMNOTAPRODIGY.IFINDITVERYDIFFICULTTOSAYWHATIMEAN--SHEOBSERVEDATLENGTH.ITSAMATTEROFTEMPERAMENT,IBELIEVE,MISSALLANHELPEDHER.THERE ARESOMEPEOPLEWHOHAVENODIFFICULTYFORMYSELFIFINDTHEREAREAGREATMANYTHINGSISIMPLYC ANNOTSAY.BUTTHENICONSIDERMYSELFVERYSLOW.ONEOFMYCOLLEAGUESNOW,KNOWSWHETHERSHELI KESYOUORNOT--LETMESEE,HOWDOESSHEDOIT--BYTHEWAYYOUSAYGOOD-MORNINGATBREAKFAST.IT ISSOMETIMESAMATTEROFYEARSBEFOREICANMAKEUPMYMIND.BUTMOSTYOUNGPEOPLESEEMTOFINDIT EASYOHNO,SAIDRACHEL.ITSHARDMISSALLANLOOKEDATRACHELQUIETLY,SAYINGNOTHINGSHESUSP ECTEDTHATTHEREWEREDIFFICULTIESOFSOMEKIND.THENSHEPUTHERHANDTOTHEBACKOFHERHEAD,A NDDISCOVEREDTHATONEOFTHEGREYCOILSOFHAIRHADCOMEIMUSTASKYOUTOBESOKINDASTOEXCUSEM E,SHESAID,RISING,IFIDOMYHAIR.IHAVENEVERYETFOUNDASATISFACTORYTYPEOFHAIRPIN.IMUS TCHANGEMYDRESS,TOO,FORTHEMATTEROFTHATANDISHOULDBEPARTICULARLYGLADOFYOURASSISTA NCE,BECAUSETHEREISATIRESOMESETOFHOOKSWHICHICANFASTENFORMYSELF,BUTITTAKESFROMTE NTOFIFTEENMINUTESWHEREASWITHYOURHELP-

5 Cipher 3: Vigenère Cipher

5.1 Identification and Period Analysis

Ciphertext 3 displayed a low IC (0.0395), characteristic of polyalphabetic substitution. To determine the key period L, I performed a Kasiski/Friedman test by calculating the average IC for substrings created by taking every L-th character. Table 3 presents the objective results of this analysis.

Period (L)	Average IC	Conclusion
1	0.03950	Random
2	0.03948	Random
3	0.03893	Random
4	0.03989	Random
5	**0.06429**	**English (Match)**
6	0.03896	Random
10	0.06459	Harmonic of 5

Table 3: Index of Coincidence for different key periods. The spike at $L = 5$ matches the expected IC for English.

The analysis objectively confirms $L = 5$ as the true period.

5.2 Cryptanalysis

The text was decomposed into 5 independent monoalphabetic streams. For each stream, I calculated the Chi-squared statistic (χ^2) against the standard English frequency distribution to find the optimal shifts statistically, avoiding manual guessing. The shifts minimizing the χ^2 value formed the key **"CVSUI"**.

5.2.1 Key Recovery Implementation

The following Python snippet demonstrates the automated logic used to derive the key by minimizing the Chi-squared statistic for each sub-stream:

```
def crack_vigenere_key(ciphertext, period):
    recovered_key = ""
    # English frequencies (A-Z,.,-,,) adapted for N=29
    ENGLISH_PROBS = {'E': 0.127, 'T': 0.091, ...}

    for i in range(period):
        # Extract the i-th column
        col = ciphertext[i::period]
        best_char = '?'
        min_chi = float('inf')

        # Test all 29 candidates
        for char_idx in range(len(ALPHABET)):
            char = ALPHABET[char_idx]
            # Decrypt column with this char
            decrypted_col_chars = []
            shift = char_idx

            for c in col:
                if c in ALPHABET:
                    p_idx = (ALPHABET.index(c) - shift) % MOD
                    decrypted_col_chars.append(ALPHABET[p_idx])
```

```
            decrypted_col = "".join(decrypted_col_chars)

            # Calculate Chi-Squared score
            score = chi_square_stat(decrypted_col)

            if score < min_chi:
                min_chi = score
                best_char = char

        recovered_key += best_char
    return recovered_key
# Result: recovered_key = "CVSUI"
```

5.3 Result

Using the recovered key `"CVSUI"`, the text was fully decrypted:

```
CANT.THINKOFTHESUNSETSANDTHEMOONRISES--IBELIEVETHECOLOURSARETHEREAREWILDPEACO
CKS,RACHELHAZARDED.ANDMARVELLOUSCREATURESINTHEWATER,HELENASSERTED.ONEMIGHTDIS
COVERANEWREPTILE,RACHELCONTINUED.THERESCERTAINTOBEAREVOLUTION,IMTOLD,HELENURG
ED.THEEFFECTOFTHESESUBTERFUGESWASALITTLEDASHEDBYRIDLEY,WHO,AFTERREGARDINGPEPP
ERFORSOMEMOMENTS,SIGHEDALOUD,POORFELLOWANDINWARDLYSPECULATEDUPONTHEUNKINDNESS
OFWOMEN.HESTAYED,HOWEVER,INAPPARENTCONTENTMENTFORSIXDAYS,PLAYINGWITHAMICROSCO
PEANDANOTEBOOKINONEOFTHEMANYSPARSELYFURNISHEDSITTING-ROOMS,BUTONTHEEVENINGOF
THESEVENTHDAY,ASTHEYSATATDINNER,HEAPPEAREDMORERESTLESSTHANUSUAL.THEDINNER-TAB
LEWASSETBETWEENTWOLONGWINDOWSWHICHWERELEFTUNCURTAINEDBYHELENSORDERS.DARKNESSF
ELLASSHARPLYASAKNIFEINTHISCLIMATE,ANDTHETOWNTHENSPRANGOUTINCIRCLESANDLINESOFB
RIGHTDOTSBENEATHTHEM.BUILDINGSWHICHNEVERSHOWEDBYDAYSHOWEDBYNIGHT,ANDTHESEAFLO
WEDRIGHTOVERTHELANDJUDGINGBYTHEMOVINGLIGHTSOFTHESTEAMERS.THESIGHTFULFILLEDTHE
SAMEPURPOSEASANORCHESTRAINALONDONRESTAURANT,ANDSILENCEHADITSSETTING.WILLIAMPE
PPEROBSERVEDITFORSOMETIMEHEPUTONHISSPECTACLESTOCONTEMPLATETHESCENE.
```

6 Source Code

All four ciphertexts were successfully identified and decrypted. The statistical indicators proved crucial in distinguishing between monoalphabetic and polyalphabetic systems. The use of automated scripts allowed for efficient frequency analysis and the execution of complex mathematical operations required for the Hill cipher.

The full source code used for this analysis, including the frequency generation tools and decryption algorithms, is available at:

`https://github.com/OrlandoTomeo/Ciphertext-Identification-and-Recovery-Challenge`