Easy Crypto

Getting in touch with cryptography

Michael Schwarz
September 13, 2016

LosFuzzy’s Training Session
1. Introduction

2. Caesar

3. Monoalphabetic substitution cipher
Introduction
Overview

We will look at some entry-level cryptography that is often used in hacklets to disguise flags.
Caesar
ILP CRWWV ZOVMLQ QOXMLKFD
*ILP CRWWV ZOVMLQ QOXFKFKD*

- Classic Caesar cipher
**ILP CRWWV ZOVMQL QOXFKFKD**

- Classic Caesar cipher
- Every letter is shifted
ILP CRWWV ZOVML QOXFKF KD

- Classic Caesar cipher
- Every letter is shifted
- If 26 letters are used: only 26 possibilities
ILP CRWWV ZOVML QOXHRKKD
JMQR DSSXW APWNRM RPYGLGLE

- Classic Caesar cipher
- Every letter is shifted
- If 26 letters are used: only 26 possibilities
- Brute force until we have meaningful text
Sample text

- Classic Caesar cipher
- Every letter is shifted
- If 26 letters are used: only 26 possibilities
- Brute force until we have meaningful text
ILP CRWWV ZOVMQL QOXFKFKD
JMQ DSXXW APWNRM RPYGLGLE
KNR ETYYX BQXOSN SQZHMHMFM
LOS FUZZY CRYPTO TRAINING

- Classic Caesar cipher
- Every letter is shifted
- If 26 letters are used: only 26 possibilities
- Brute force until we have meaningful text
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter

- Caesar cipher is a “systematic” monoalphabetic substitution cipher

- Higher complexity: polyalphabetic substitution cipher
  - One *symbol* is made of multiple letters
  - Vigenere cipher is a “systematic” polyalphabetic substitution cipher
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter
  - Each letter is replaced by a different letter

Caesar cipher is a “systematic” monoalphabetic substitution cipher

Vigenere cipher is a “systematic” polyalphabetic substitution cipher
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter
  - Each letter is replaced by a different letter
  - Caesar cipher is a “systematic” monoalphabetic substitution cipher
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter
  - Each letter is replaced by a different letter
  - Caesar cipher is a “systematic” monoalphabetic substitution cipher
- Higher complexity: polyalphabetic substitution cipher
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter
  - Each letter is replaced by a different letter
  - Caesar cipher is a “systematic” monoalphabetic substitution cipher
- Higher complexity: polyalphabetic substitution cipher
  - One *symbol* is made of multiple letters
Substitution Cipher

- Replace *symbols* in plaintext with different *symbols*
- Simplest case: monoalphabetic substitution cipher
  - One *symbol* is one letter
  - Each letter is replaced by a different letter
  - Caesar cipher is a “systematic” monoalphabetic substitution cipher
- Higher complexity: polyalphabetic substitution cipher
  - One *symbol* is made of multiple letters
  - Vigenere cipher is a “systematic” polyalphabetic substitution cipher
Monoalphabetic substitution cipher
AX qgm lZafc UjqhlgYjShZq ak IZW SfkoWj lg qgmj hjgTdWe, IZWf qgm Vgf2l cfg0 oZSl qgmj hjgTdWe ak. FWmeSff, usst
AX qgm lZafc UjqhlgYjShZq ak lZW SfkoWj lg qgmj hjgTdWe, lZWf qgm Vgf2l cftgo oZSI qgmj hjgTdWe ak. FWmeSff, usst

- More fun if the alphabet size is unknown
CAesar

AX qgm lZafc Ujqh1gYjShZq ak lZW SfkoWj lg qgmj hjgTdWe, lZWf qgm Vgf2l cfgo oZSl qgmj hjgTdWe ak. FWmeSff, usst

- More fun if the alphabet size is unknown
- Still easy to brute force
Caesar brute force - Code

```python
upper = "".join([chr(ascii) for ascii in range(65,91)])
lower = "".join([chr(ascii) for ascii in range(97,123)])
digit = "".join([chr(ascii) for ascii in range(48,58)])

# select the alphabet: uppercase, lowercase, digits and apostrophe
ALPHABET = upper + lower + digit + "'"
# try all possible keys
KEY = range(len(ALPHABET))
# message to decode
MSG = "AX qgm lZafc UjqhlgYjShZq ak lZW SfkOWj lq qgmj" +
    "hjgTdWe, lZWf qgm Vgf2l cfgO oZSl qgmj hjgTdWe ak. FWmeSff, usst"

for k in KEY:
    out = ""
    for c in MSG:
        try:
            out += ALPHABET[(ALPHABET.index(c) + k) % len(ALPHABET)]
        except:
            out += c
    print("%d: %s" % (k, out))
```
Caesar brute force

If you think cryptography is the answer to your problem, then you don’t know what your problem is. Neumann, 2001
Good news everyone

we are done with brute force
Substitution cipher

WSie ueScADSGf
j bHcf heSifSD i hNNB YeScSGfifANG ONe fsS zNc
mHIIkc. uBSicS OAGD Af iffihsSD fN fsAc StiAB. j
SGhekYfSD Af EAfs iSc HcAGR i 128-KAf qSk
RSGSeifSD HcAGR fsS eiGD() OHGhfANG NO fsS h
cfiGDieD BAKEiek. PNfs fsif j hNtYABSD fsS qSk
fNDidk if 12:00it.
jG iDDAfANG fN fsAc YeScSGfifANG, j iDDSD iG
iefAhBS iKNHf iG AGfSeScfAGR hekYfN-iBRNeAfst.
LINESQSe, AG NeDSe fN eSiD fsS iefAhBS (Af Ac
iYYeNwAtifSBk 3500 hsieihfSec BNGR), kNH GSSD fN
eSQSecS Esif j DAD fN sADS Af’c hNGfSGf OeNt kNH.
j RAQS kNH i sAGf: ZNH’BB GSSD fsS sSBY NO i
OitNHC eNtiG StYSeNe.
lsSSSec, xBAhS
Substitution cipher - Most frequent letter S to e

Weie ueecADeGf
j bHcf heeifeD i hNNB YeeceGfifANG ONe fse zNc
mHIIkc. uBeice OAGD Af iffihseD fn fsAc etiAB. j
eGhekYfeD Af EAfs iec HcAGR i 128-KAf qek
ReGeeifeD HcAGR fse eiGD() O HGhfANG NO fse h
cfiGDieD BAKEieek. PNfe fsif j hNtYABeD fse qek
fNDik if 12:00it.
jG iDDAfANG fn fsAc YeeceGfifANG, j iDDDe iG
iefAhBe iKNHf iG AGfeeeefcAGR hekYfN-iBRNeAfst.
LNEeQee, AG NeDee fn eeID fse iefAhBe (Af Ac
iYYeNwAtifeBk 3500 hsieihfeec BNGR), kNH GeeD fn
eeQeece Esif j DAD fn sADe Af’c hNGfeGf OeNt kNH.
j RAQe kNH i sAGf: ZNH’BB GeeD fse seBY NO i
OitNHc eNtiG etYeeNe.
lseeec, xBAhe
Weie ueecADeGf
j bHcf heeifeD i hNNB YeeceGfifANG ONe fse zNc
mHIckc. uBeice OAGD Af iffihseD fN fsAc etiAB. j
eGhekYfeD Af EAfA iec HcAGR i 128-KAf qek
ReGeeifeD HcAGR fse eiGD() OHGhfANG NO fse h
cfiGDieD BAKeiek. PNfe fsif j hNtYABeD fse qek
fNDik if 12:00it.
jG iDDAfANG fN fsAc YeeceGfifANG, j iDDeD iG
iefAhBe iKHf iG AGfeeeecfAGR hekYfN-IBRNeAfst.
LNEeQee, AG NeDee fN eeID fse iefAhBe (Af Ac
iYYeNwAtifeBk 3500 hsieihfecd BNGr), kNH GeeD fN
eQeece Esif j DAD fN sADe Af’c hNGfeGf OeNt kNH.
j RAQe kNH i sAGf: ZNH’BB GeeD fse seBY NO i
OitNHC eNtiG etYeeNe.
lseeec, xBAhe
Weir uresADEGf
j bHsf hreifeD i hNNB YreseGfifANG ONr fhe zNs
mHIIks. uBeise OAGD Af iffihheD fN fhAs etiAB. j
eGhrkYfeD Af EAfh ies HsAGR i 128-KAf qek
ReGerifeD HsAGR fhe riGD() OHGhfANG NO fhe h
sfiGDiRD BAKrirk. PNfe fhif j hNtYABeD fhe qek
fNDik if 12:00it.
jG iDDAfANG fN fhAs YreseGfifANG, j iDDDe iG
irfAhBe iKNHf iG AGferesfAGR hrkYfN-IBRNrAfht.
LNEeQer, AG NrDer fN reID fhe irfAhBe (Af As
iYYrNwAtifeBk 3500 hhirihfes BNGR), kNH GeeD fN
reQerse Ehif j DAD fN hADE Af’s hNGfeGf OrNt kNH.
j RAQe kNH i hAGf: ZNH’BB GeeD fhe heBY NO i
OitNHs rNtiG etYerNr.
cheers, xBAhe
Weir uresADeGf
j bHsf hreifeD i hNNB YreseGfifANG ONr fhe zNs
mHIIks. uBeise OAGD Af iffihheD fN fhAs etiAB. j
eGhrkYfeD Af EAfh ies HsAGR i 128-KAf qek
ReGerifeD HsAGR fhe riGD() OHGhfANG NO fhe h
sfiGDirD BAKrirk. PNfe fhif j hNtYABeD fhe qek
fNDik if 12:00it.
jG iDDAfANG fN fhAs YreseGfifANG, j iDDeD iG
irfAhBe iKNHf iG AGferesfAGR hrkYfN-iBRNrafht.
LNEeQer, AG NrDer fN reiD fhe irfAhBe (Af As
iYYrNwAtifeBk 3500 hhirihfers BNGR), kNH GeeD fN
reQerse Ehif j DAD fN hADE Af’s hNGfeGf OrNt kNH.
j RAQe kNH i hAGf: ZNH’BB GeeD fhe heBY NO i
OitNhs rNtiG etYerNr.
cheers, xBAhe
Weir uresADeGt
j bHst hreiteD i hNNB YreseGtitANG ONr the zNs
mHIIkS. uBeise OAGD At ittihhheD tN thAs etiAB. j
eGhrkYteD At EAth ies HsAGR i 128-KAt qek
ReGeriteD HsAGR the riGD() OHGhtANG NO the h
stiGDirD BAKrirk. PNte thit j hNtYABeD the qek
tNDik it 12:00it.
jG iDDAtANG tN thAs YreseGtitANG, j iDDed iG
irtAhBe iKNHt iG AGterestAGR hrkYTn-iBRNraht.
LNEeQer, AG NrDer tN reI D the irtAhBe (At As
iYYrNWAtiteBk 3500 hhirihters BNGR), kNH GeeD tN
reQerse Ehit j DAD tN hADe At’s hNGteGt OrNt kNH.
j RAQe kNH i hAGt: ZNH’BB GeeD the heBY NO i
OitNHs rNtiG etYerNr.
cheers, xBAhe
Substitution cipher - Guess words

Weir uresiDeGt
j bHst hreiteD i hNNB YreseGtitiNG ONr the zNs
mHIIk. uBeise OiGD it ititiheD tN this etiiB. j
eGhrkYteD it Eith ies HsiGR i 128-Kit qek
ReGeriteD HsiGR the riGD() OHGhtiNG NO the h
stiGDirD BiKrirk. PNte thit j hNtYiBeD the qek
tNDik it 12:00it.
jG iDDiING tN this YreseGtitiNG, j iDDDe iG
irtihBe iKNHt iG iGterestiGR hrkYtN-iBRNritht.
LNEeQer, iG NrDer tN reiD the irtihBe (it is
iYYrNwiteitBk 3500 hhirihters BNGR), kNH GeeD tN
reQerse Ehit j DiD tN hiDe it’s hNGteGt OrNt kNH.
j RiQe kNH i hiGt: ZNH’BB GeeD the heBY NO i
OitNHS rNtiG etYerNr.
cheers, xBihe
Substitution cipher - Guess words

Weir uresiDeGt
j bHst hreiteD i hNNB YreseGtitiNG ONr the zNs
mHIiks. uBeise OiGD it ittihheD tN this etiiB. j
eGhrkYteD it Eith ies HsiGR i 128-Kit qek
ReGeriteD HsiGR the riGD() OHGhtiNG NO the h
stiGDirD BiKirk. PNte thit j hNtYiBeD the qek
tNDik it 12:00it.
jG iDDitiNG tN this YreseGtitiNG, j iDDed iG
irtihBe iKNHt iG iGterestiGR hrkYtN-iBRNritht.
LNEeQer, iG NrDer tN reID the irtihBe (it is
iYYrNwititeBk 3500 hhiirihters BNGR), kNH GeeD tN
reQerse Ehit j DiD tN hiDe it’s hNGteGt OrNt kNH.
j RiQe kNH i hiGt: ZNH’BB GeeD the heBY NO i
OitNHs rNtiG etYerNr.
cheers, xBihe
Weir uresiDent

j bHst hreiteD i hNNB YresentitiNn ONr the zNs mHIIks. uBeise OinD it ittihheD tN this etiiB. j enhrkYteD it Eith ies Hsing i 128-Kit qek generiteD Hsing the rinD() OHnhtiNn NO the h stinDirD BiKirk. PNte thit j hNtYiBeD the qek tNDik it 12:00it.

jn iDDitiNn tN this YresentitiNn, j iDDeD in irtihBe iKNHt in interesting hrkYtN-iBgNritht. LNEeQer, in NrDer tN reiD the irtihBe (it is iYYrNWititeBk 3500 hhirihters BNng), kNH neeD tN reQerse Ehit j DiD tN hiDe it’s hNntent OrNt kNH. j giQe kNH i hint: ZNH’BB neeD the heBY NO i OitNHS rNtin etYerNr.

cheers, xBihe
Weir uresiDent
j bHst hreiteD i hNNB YresentitiNn ONr the zNs mHIIks. uBeise OinD it ittihheD tN this etiiB. j enhrkYteD it Eith ies Hsing i 128-Kit qek generiteD Hsing the rinD() OHnhtiNn NO the h stinDirD BiKrirk. PNte thit j hNtYiBeD the qek tNDik it 12:00it.
jn iDDitiNn tN this YresentitiNn, j iDDDeD in irtihBe iKNHt in interesting hrkYtN-iBgNritht. LNEeQer, in NrDer tN reiD the irtihBe (it is iYYrNwititeBk 3500 hhirihters BNng), kNH neeD tN reQerse Ehit j DiD tN hiDe it’s hNntent OrNt kNH. j giQe kNH i hint: ZNH’BB neeD the heBY NO i OitNHS rNtin etYerNr.
cheers, xBihe
Substitution cipher - Guess words

Wear uresiDent
j bHst hreateD a hooB presentation Oor the zos mHIIIks. uBease OinD it attahheD to this etaiB. j enhrkpteD it Eith aes Hsing a 128-Kit qek generateD Hsing the ranD() OHnhtion oO the h stanDarD BiKrark. Pote that j hotpiBeD the qek toDak at 12:00at.
jn aDDition to this presentation, j aDDeD an artihBe aKоФt an interesting hrkpto-aBgoritht. LoEeQer, in orDer to reaD the artihBe (it is approwitateBk 3500 hharahters Bong), koH neeD to reQerse Ehat j DiD to hiDe it’s hontent Orot koH. j giQe koH a hint: ZoH’BB neeD the heBp oO a OatoHs rotan etperor.
cheers, xBihe
Substitution cipher - Guess words

Wear uresiDent
j bHst hreateD a hooB presentation Oor the zos
mHIIks. uBease OinD it attahheD to this etaiB. j
enhrkpteD it Eith aes Hsing a 128-Kit qek
generateD Hsing the ranD() OHnhtion oO the h
stanDarD BiKrark. Pote that j hotpiBeD the qek
toDak at 12:00at.

jn aDDDition to this presentation, j aDDDeD an
artihBe aKoHt an interesting hrkppto-aBgorithht.
LoEeQer, in orDer to reaD the artihBe (it is
approwitateBk 3500 hharahters Bong), koH neeD to
reQerse Ehat j DiD to hiDe it’s hontent Orot koH.
j giQe koH a hint: ZoH’BB neeD the heBp oO a
OatoHs rotan etperor.

cheers, xBihe
Wear uresident
j bHst hreated a hooB presentation Oor the zos mHIIks. uBease Oind it attahhed to this etaiB. j enhrkpted it Eih aes Hsing a 128-Kit qek generated Hsing the rand() OHnhtion oO the h standard BiKrark. Pote that j hotpiBed the qek todak at 12:00at.
jn addition to this presentation, j added an artihBe aKoHt an interesting hrkpto-aBgorithht. LoEeQer, in order to read the artihBe (it is approwitateBk 3500 hharahahters Bong), koH need to reQerse Ehat j did to hide it’s hontent Orot koH. j giQe koH a hint: ZoH’BB need the heBp oO a OatoHs rotan etperor.
cheers, xBihe
Wear uresident

j bHst hreated a hooB presentation Oor the zos mHIiks. uBease Oind it attahhed to this etaiB. j enhrkpted it Eith aes Hsing a 128-Kit qek generated Hsing the rand() OHnhtion oO the h standard BiKrark. Pote that j hotpiBed the qek todak at 12:00at.

jn addition to this presentation, j added an artihBe aKoHt an interesting hrkpto-aBgoritht. LoEeQer, in order to read the artihBe (it is approwitateBk 3500 hharahters Bong), koH need to reQerse Ehat j did to hide it’s hontent Orot koH. j giQe koH a hint: ZoH’BB need the heBp oO a OatoHs rotan etperor.

cheers, xBihe
Dear uresident
I bust created a cool presentation for the zos muIIys. ulease find it attached to this email. I encrypted it with aes using a 128-Kit qey generated using the rand() function of the c standard liKrary. Note that I compiled the qey today at 12:00am.

In addition to this presentation, I added an article aKout an interesting crypto-algorithm. LoweQer, in order to read the article (it is approxowimately 3500 characters long), you need to reQerse what I did to hide it’s content from you. I giQe you a hint: You’ll need the help of a famous roman emperor.
cheers, xlice
Dear President
I just created a cool presentation for the Los Fuzzys. Please find it attached to this email. I encrypted it with aes using a 128-bit key generated using the rand() function of the c standard library. Note that I compiled the key today at 12:00am.
In addition to this presentation, I added an article about an interesting crypto-algorithm. However, in order to read the article (it is approximately 3500 characters long), you need to reverse what I did to hide it’s content from you. I give you a hint: You’ll need the help of a famous roman emperor.
Cheers, Alice
TOO EASY

WHAT'S NEXT?
Vigenère
Vigenère

• Polyalphabetic substitution cipher
• Composed of multiple Caesar ciphers
• Uses a keyword to select the Caesar shift
Vigenère

- Polyalphabetic substitution cipher
Vigenère

- Polyalphabetic substitution cipher
- Composed of multiple Caesar ciphers
Vigenère

- Polyalphabetic substitution cipher
- Composed of multiple Caesar ciphers
- Uses a keyword to select the Caesar shift
**LOS FUZZYS**

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |

**KEY**

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |

**VSQ _____**
LOS FUZZYS

VSQ P_____
**LOS FUZZYS**

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |

**KEY**

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |

**VSQ PY**
LOS FUZZYS

VSQ PYX
**LOS FUZZYS**

|   | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| A |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| B |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| C |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| D |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| E |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| F |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| G |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| H |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| I |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| J |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| K |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| L |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| M |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| N |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| O |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| P |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| Q |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| R |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| S |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| T |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| U |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| V |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| W |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| X |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| Y |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |
| Z |   | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |   |

**KEY**

- **VSQ PYXJC**
LOS FUZZYS

KEY

VSQ PYXJCQ
How to break

- Finding the key length using the Kasiski examination:
How to break

• Finding the key length using the Kasiski examination:
  • Find same words
How to break

- Finding the key length using the Kasiski examination:
  - Find same words
  - Note the difference in their text position
How to break

- Finding the key length using the Kasiski examination:
  - Find same words
  - Note the difference in their text position
  - Factor all differences
How to break

• Finding the key length using the Kasiski examination:
  ● Find same words
  ● Note the difference in their text position
  ● Factor all differences
  ● The factor that occurs most often is probably the key length
Kasiski examination

myx iyhe,

h gcflc xmx qzmr yi ings or zli gzjc xild kthdx. rt gzjc xoqd wto zqc xohsygfd emw ngd htv vd ufhs sm ryds wto. cn wto jmmb ngd ajhsqyq jzqi xnzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl uq? okcfmd lypy rtpj snt bth’s cpfq zsrjhshms nn xmxlrdjk qghjj snt’pj qzhrnhf emw or.

vynn nm y gymbf fn okyyznqk xyudl, by vhjq zhmb dit sfjld. emqfnv sx omnzyltrgaykx!

yx nghq jgzhj nm rspthfkw jhbqwundc g hum optphcc dit vgyb lnpj cmemwgzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmgllnnm. rmy okys cr sm fnszap xtagsa ktlhb shkj vdbyzmd lyss dlnqixdcx qhkj gy ntr yujhll ngdgw ftmam cm sfj ldrrfoqzlym hm rmy mdglbanpminc. uj uqd etcmf rt ord rmy azap ymspfhbd ysx adffpd kgpy rtnufhdpx ie emtx. nmaj qd zpj cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknqj nn sfj wdmrwuk uyzfs. hr nm oqmyybsc vx rcayqzj lozqbx umc y mcfgjd mnofnmshafndc jfmsdp xsrsocr. nn jltwj nsy ngd ezuqcq by vhjq ord afmfnqmkql. rt vqhl ll ngd kthdx mzn ne rmy azlp qd vgqf trc f znmb ylnkjd.

qd bys xhrazmr dtjlxsfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,
h gcflc xmpz qzmr yi ings or zli gzjc xild kthdx. rt gzjc xoqd wto zqc
xohsygfd emw ngd htv vd ufhs sm ryds wto. cn wto jmb ngd ajhsqyq jzqi
xnzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl ug? okcfmd lpy
rtpj snt bth’s cpfq zsrjhsms nn xmrldjk qghj jnt’pj qzhrnhf emw or.
vynn nm y gymbf fn okyyznqk xyudl, by vhjx zhmb dit sfjld. emqfnv sx
omnzyltrgaykt!
yx nghq jghzj nm rsptfhkw jhbqwundc g hum optphcc dit vgb lnpj
cmemwggzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmgllnm. rmy
okys cr sm fnszap xqgsa kthhb shkj vdbzymd lyss dlnqsdcx qhkj gy ntr
yujhh ngdhw ftmam cm sfj ldrhfoqzlym hm rmy mdglbanpmnc. uj uqd etcmf
rt ord rmy azap ymshfbd ysx adffpd kgpy rtnufhdpv ie emtx. nmaj qd zpj
cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknqj nn sfj wdhrwuk uyzfs.
hr nm oqmyybsci vx rcayqzj losqzx umc y mcfgjd mnofnmsffndc jfmsdp
xsrsr. nn jltwj nsg ngd ezuqcx by vhjx ord amfnqmkqi. rt vqhl ngd
kthdx mzn ne rmy azlp qd vgqw trc f znnb ylnkjd.
qd bys xhrazmr dtjxlzfnhj djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,

h gcflc xmx zmasm or yzv gzf jxld kthdx. rt gzf xoqd wto zqczxosygfd emw ngd htv vd ufhs sm ryds wto. cn wto jmmngd ajhgsqvq jzqcxnzsgth? zqczit ztfckzzqy nm kthczw scmdrijmsf, xyudl ug? okcfmd lypy rtpj snt bth’s cpfq zsrjshsms nn xmrzldjkl qghjj snt’pj qzhrnhf emw or.
yynn nm y gyymbf fn okyyznqk xyudl, by vhjq zhmb dit sfjld. emqfnv sx omnzyltrgyx!
yx nghq jgzhj nm rs nthfkw jhbqwundc g hum optphcc dit vgyb lnpjcmemwgzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmglllnnm. rmy okys cr sm fnszap xtgasa ktlhb shkj vbdbyzmd lyss dlgnqixdxc qhkj gy ntr yujhllngdgw ftmam cm sfj ldrرفقزليم hm rmy mdglbanpminc. uj uqd etcmf rt ord rmy azap ymzpfd kgpy rtnufhpdx ie emtx. nmaj qd zpjcm sfj vthjicmf uj qhkj zmd nsw wnsuc wnnfcy wknqj nn sfj wdmwrwuk uyzvs.
hrm oqmyybsci vx rcayqzj lorraine umc y mcfgjdmn fmshafndc jfmsdp xsrscr. nn jltwj nsy ngd ezuqcy by vhjq ord amfnqmkqj1. rt vqhl ngd kthdx mzn ne rmy azlp qd vgqf trc f znnb ylnkjd.
qd bys xhrzmr dtjlxsfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,

h gcflc xmx qzmr yi ings or zli gzjc xild ktxhd. rt gzjc xoqd wto qzc
xohsygfq emw ngd htv vd ufhs sm ryds wto. cn wto jmmw ngd ajshsqyq jzqi
xznzsgth? qzc dt ztfckzzqy nm kthczw scmdrymsf, xyudl ug? okcfmd lpy
rtpj snt bth's cpfq zsyrjshms nn xtmzldjxk qghhj snt'pj qzhrnhf emw or.
vynn nm y gymbf fn okyyznk xyudl, by vhjx zhmb dit sfdj. emqfnv sx
omznzylrkgkx!

yx nghq jgzhj nm rspsyfkw jhbqwundc g hum optphcc dit vgyb lnpj
cmemwgszsgth: ntp yuqfyc qhkj gy sgc sushmsuk ayse ne umfgnhlnnm. rmy
okys cr sm fnzsp xtnqsa ktlhb shkj vdbyzmd lyss dlnqixdcx qhkj gy ntr
yujhll ngdgg ftmam cm sfj ldrfogzlym hm rmy mdglbanpmnc. uj uqd etcmef
rt ord rmy azap ymspfhbd ysx adffpd kgpy rtmufhdpx ie emtx. nmaj qd zpj
cm sfj vthjcminf uj qhkj zmd nsw wnucw nn fcy wknnqj nn sfj wdmrwik uyzfs.
hr nm oqmyybscix vx rcayqqz lozqbx umc y mcflj elnognmshfndc jflsdp
xsrsbry. nn jltwy nsy ngd ezuqcy by vhjx ord amfnqmkqi1. rt vqhlx ngd
ktxhd mzn ne rmy azlp qd vgqf trc f znnb ylnkjd.
qd bys xhrazmr dtjlxsfnhf djxy nm kthcww! ho, anz
Kasiski examination

myx iyhe,

h gcflc xmx qzmr yi ings or zli gzjc xild kthdx. rt gzjc xoqd wto zqc
xohsygfld emw ngd htv vd ufhs sm ryds wto. cn wto jmmb ngd ajhsqyq jzqi
xnzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl ug? okcfmd lppy
rtpj snt bth’s cpfq zsrjhshms nn xmxldrjyk qghjj snt’pj qzhrnhf emw or.
vynn nm y gymbf fn okyyznqk xyudl, by vhjq zhmb dit sfjld. emqfnv sx
omnzyltrgaykx!

yx nghq jgzhj nm rsptfkw jhbqwundc g hum optphcc dit vgyb lnpj
cmemwzgzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmgllnmm. rmy
okys cr sm fnzsap xtqgsa ktlhb shkj vdbyzmd lyss dlnqixdcx qhkj gy ntr
yujhll ngdgw ftmam cm sfj ldrfoqzlym hm rmy mdglbanpminc. uj uqd etcmf
rt ord rmy azap ymsspfd ky py rtnufhdpzx ie emtx. nmaj qd zpj
cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknqj nn sfj wdmrwuk uyzfs.
hr nm oqmyybsci vx rcayqzj loqsbx umc y mcfgjd mnofnmshaftndc jfmsdp
xsrsr. nn jltwj nsy ngd ezuqcz by vhjq ord amfnqmkjql. rt vqhlld ngd
kthdx mzn ne rmy azlp qd vggf trc f znnb ylnkjd.
qd bys xhraznr dtjlxsfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,

h gcflc xmx qzmr yi ings or zli gzjc xild kthdx. rt gzjc xoqd wto zqc xohsygfd emw ngd htv vd ufhs sm ryds wto. cn wto jmmmb ngd ajhsqyq jzqi xnzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl ug? okcfmd lppy rtpj snt bth’s cpfq zsrjhshms nn xmxlrdjk qghj jn tpj qzhrnhf emw or. vynn nm y gymbf fn okyyzngk xyudl, by vhjq zhmb dit sfjld. emqfnv sx omnzyltrgaykx!

yx nghq jgzhj nm rsptthfw kw jhbwundc g hum optphcc dit vgyb lnpj cmemwqzgsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmgllnnm. rmy okys cr sm fnzwap xtqgsa ktlhb shkj vdbyzmd lyss dlnqixdcx qhkj gy ntr yujhll ngdgw ftmam cm sfj ldrfogzlym hm rmy mdglbanpminc. uj uqd etcmf rt ord rmy azap ymspfhbd ysx adffpd kgpy rtnufhdpx ie emtx. nmaj qd zpj cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknqj nn sfj wdmrwuk uyzfs.

hr nm oqmyybsci vx rcayqzj lozqbx umc y mcfgjd mnofnmsafndc jfmsdp xsrsccr. nn jltwj nsy ngd ezuqcq by vhjq ord amfnqmkiql. rt vqhl ngd kthdx mzn ne rmy azlp qd vgqf trc f znnb ylnkjd.

qd bys xhrazmr dtjlxsfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,

h gcflc xmxqzmr yiyings or zli gzjcxild kthdx. rt gzjcxoqd wto zqc
xohsygyfd emw ngd hvvd ufhs sm ryds wto. cn wto jmmmb ngd ajhsqyq jzqi
xnzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl ug? okcfmd lypy
rtpj snrtbths cpfq zsrjshshms nn xmzlrdjk qghjj sntrpj qzhrnhf emw or.
vynn nm y gymbf fn okyyznqk xyudl, by vhjq zhmb dit sfjld. emqfnv sx
omnzyltrygaykx!

yx nghq jgzhj nm rsptthfkw jhbqwundc g hum optphcc dit vgyb lnpj
cmemwzgsgth: ntp yuqfcy qhjk gy sgc sushmsuk ayse ne ufmgkhllnnm. rmy
okys cr sm fniszapot qgsga kthhb shkj vdbyzmd lyss dlnqixdctx qhjk gy ntr
yujhlh ngdgw ftmam cm sfj ldrrfoqzlym hm rmy mdgblanpmnc. uj uqd etcmf
rt ord rmy azap ympfsfbd yxs adffpd kgpy rtnufrdpx i e emtx. nmaj qd zpj
cm sfj vthjicmf uj qhjkj zmd nsw wnucc nn fcy wknqj nn sfj wdmrwuk uyzfs.
hm nqomqmyybsci vx rccayqz jzloqblx umc y mcrgjd mnofnnshafndc jfmsdp
xsrsiccr. nn jltwj nsy ngd ezuqcq by vhjq ord amfnqmkqil. rt vqhlh ngd
kthdx mzn ne rmy azlp qd vgqf trf znnb ylnkjd.
qd bys xhrayzmr dtjilxfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,
h gcflc xmx qzmr yi ings or zli gzjc xild kthdx. rt gzjc xoqd wto zqc xohsygfdr emw ngd htv vd ufhs sm ryds wto. cn wto jjmm hlg ajhsqyg jzqi xznzsgth? zqc dit ztfckzzqy nm kthczw scmdrjymsf, xyudl ug? okcfmd lppy rtpj snt bth’s cpfq zsrjhshms nn xzmzlrjdk qghj jnt’pj qzhrnhf emw or. vynn nm y gymbf fn okyyznng xuydl, by vhjq zhmb dit sfjld. emqfnv sx omnzyltrgaykx!
yx nghq jgzhj nm rsptfhkw jhbqwundc g hum optphcc dit vgyb lnppj cmemwgnzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmgllinnm. rmy okys cr sm fnzmp xtqgsa ktlhb shkj vdbysmd lyss dlnqixdcx qhkj gy ntr yujhll ngdgw ftmam cm sfj ldrhrqzlym hm rmy mdglnbpmnc. uj uqdf etcmtf rt ord rmy azap ymzpflhdys xsy adeffpogkgpy rtrnufhdpj ie emtx. nmaj qd zpj cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknnj nn sfj wdmrwuk uzysf. hr nm qomnybybci vx rcayqjz lozqbx umc y mcggjy mnofnmsnhafndc jfmsdp xsrsr. nn jltwj nsy ngd ezuqcq by vhjq ord amfqnkmkiql. rt vqhlh ngd kthdx mzn ne rmy azlp qd vggqf trc f znnb ylnkjd.
qd bys xhrazmr dtjlxsfnhf djxy nm kthczw! ho, anz
Kasiski examination

myx iyhe,
h gcflc xmx zqmr yi ings or zli gzjc xil f kthdx. rt gzjc xoqd wto zqc
xohsygfd emw ngd htv vd ufhs sm ryds wto. cn wto jmmmb ngd ajhsqyq jzqi
xnzsgth? zqc dit ztfckzzqy nm kthcww scmdrjymsf, xyudl ug? okcfmd lypy
rtpj snt bth’s cpfq zsrjhsms nn xmslrdjk qghjj snt’pj qzhrnhf emw or.
vynn nm y gymbf fn okyyznqk xyudl, by vhjq zhmh dd sfjld. emqfnv sx
omnzyltrgaykx!

yx nghq jgzhj nm rsptfkw jhbqwundc g hum optphcc dit vgyb lnpj
cmemwgzsgth: ntp yuqfcy qhkj gy sgc sushmsuk ayse ne ufmglllnnm. rmy
okys cr sm fnszap xtmpsa ktlhb shkj vdbyzmd lyss dlnqixdctx qhkj gy ntr
yujlhl ngdgw ftmam cm sfj ldrrfoqzlym hm rmy mdglbanpmnc. uj uqd etcmf
rt ord rmy azap ymspfhbd ysx adffpd kgpy rtnufhdpx ie emtx. nmaj qd zpj
cm sfj vthjicmf uj qhkj zmd nsw wnucw nn fcy wknqj nn sfj wdmrwuk uyzfs.
hr nm oqmyybsci vx rcayqzj lozqbx umc y mcggjd mnofsmshafndc jfmsdp
xsrsca. nn jltwv jsy ngd ezuqcq by vhjq ord amfqnmlqy. rt vqhl yngd
kthdx mzn ne rmy azlp qd vqgf trc f znnb ylnkjd.
qd bys xhrazmr dtjlxsfnhf djxy nm kthcww! ho, anz
Finding the key length
Finding the key length

![Bar chart showing key length frequencies]

- Key length 4: 1 count
- Key length 5: 15 counts
- Key length 6: 4 counts
- Key length 7: 1 count
- Key length 8: 1 count
- Key length 9: 1 count
- Key length 10: 1 count
- Key length 11: 1 count
Continue breaking the cipher

• We (hopefully) know the key length
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
- In this case: key length of 5 means we have 5 Caesar ciphers
● We (hopefully) know the key length
● We can disassemble the cipher text into multiple Caesar ciphers
● In this case: key length of 5 means we have 5 Caesar ciphers
● 1., 6., 11., ... letter is encrypted using a Caesar ciphers
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
- In this case: key length of 5 means we have 5 Caesar ciphers
- 1., 6., 11., ... letter is encrypted using a Caesar ciphers
- 2., 7., 12., ... letter is encrypted using a Caesar ciphers
• We (hopefully) know the key length
• We can disassemble the cipher text into multiple Caesar ciphers
• In this case: key length of 5 means we have 5 Caesar ciphers
• 1., 6., 11., ... letter is encrypted using a Caesar ciphers
• 2., 7., 12., ... letter is encrypted using a Caesar ciphers
• 3., 8., 13., ... letter is encrypted using a Caesar ciphers
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
- In this case: key length of 5 means we have 5 Caesar ciphers
- 1., 6., 11., ... letter is encrypted using a Caesar cipher
- 2., 7., 12., ... letter is encrypted using a Caesar cipher
- 3., 8., 13., ... letter is encrypted using a Caesar cipher
- 4., 9., 14., ... letter is encrypted using a Caesar cipher
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
- In this case: key length of 5 means we have 5 Caesar ciphers
- 1., 6., 11., ... letter is encrypted using a Caesar cipher
- 2., 7., 12., ... letter is encrypted using a Caesar cipher
- 3., 8., 13., ... letter is encrypted using a Caesar cipher
- 4., 9., 14., ... letter is encrypted using a Caesar cipher
- 5., 10., 15., ... letter is encrypted using a Caesar cipher
Continue breaking the cipher

- We (hopefully) know the key length
- We can disassemble the cipher text into multiple Caesar ciphers
- In this case: key length of 5 means we have 5 Caesar ciphers
- 1., 6., 11., ... letter is encrypted using a Caesar cipher
- 2., 7., 12., ... letter is encrypted using a Caesar cipher
- 3., 8., 13., ... letter is encrypted using a Caesar cipher
- 4., 9., 14., ... letter is encrypted using a Caesar cipher
- 5., 10., 15., ... letter is encrypted using a Caesar cipher
- Use frequency analysis on each Caesar cipher
Frequency analysis of first Caesar (1., 6., 11., ...)

- Most frequent letter: J
Frequency analysis of first Caesar (1., 6., 11., ...)

- Most frequent letter: J
- Guess that J equals E in original text
Frequency analysis of first Caesar (1., 6., 11., ...)

- Most frequent letter: J
- Guess that J equals E in original text
- Looking at column E until we find J gives first letter of key: F
Frequency analysis of first Caesar (2., 7., 12., ...)

- Most frequent letter: Y

Guess that Y equals E in original text.

Looking at column E until we find Y gives second letter of key: U.
Frequency analysis of first Caesar (2., 7., 12., ...)

- Most frequent letter: Y
- Guess that Y equals E in original text
Frequency analysis of first Caesar (2., 7., 12., ...)

- Most frequent letter: Y
- Guess that Y equals E in original text
- Looking at column E until we find Y gives second letter of key: U
Frequency analysis of first Caesar (3., 8., 13., ...)

- Most frequent letter: **N**
Most frequent letter: N

Guess that N equals E in original text
Frequency analysis of first Caesar (3., 8., 13., ...)

- Most frequent letter: N
- Guess that N equals E in original text
- Looking at column E until we find N gives third letter of key: J
Frequency analysis of first Caesar (4., 9., 14., ...)

- Most frequent letter: D
• Most frequent letter: D
• Guess that D equals E in original text
Frequency analysis of first Caesar (4., 9., 14., ...)

- Most frequent letter: **D**
- Guess that **D** equals **E** in original text
- Looking at column **E** until we find **D** gives fourth letter of key: **Z**
Frequency analysis of first Caesar (5., 10., 15., ...)

- Most frequent letters: C and Y
Frequency analysis of first Caesar (5., 10., 15., ...)

- Most frequent letters: C and Y
- Guess that C or Y equals E in original text
Frequency analysis of first Caesar (5., 10., 15., ...)

- Most frequent letters: C and Y
- Guess that C or Y equals E in original text
- Looking at column E until we find C or Y gives fifth letter of key: Y or U
heo jack,

y heart you wqnt to zoin ui and mqke soce monuy. to mqke suhe you qre suytablu for txe job me wanj to meut you. to you anow txe cenjral pqrk stqtion? qre yok avaibable en montay nideteedth, selen pm? fleasu make iure yeu don’j draw qttenjion te yourielf wxile you’re wqitinw for ui. wait en a bedch at flatferm selen, we mill fynd yok theru. follew us udobtrksiveby!

as txis emqil is itronwly ensryptud i cad provyde yok with core idformqtion: eur tahget wyll be jhe najionab bank ef wasxingten. the flan ii to atjack dkring bunch jime bucausu many umplooees wyll be eut taaing txeir lknch id the rustauhants yn the deighrorhoed. we ahe goidg to uie the rack edtranse and rehavu like iupplyers ov food. ence wyll usu our cever te get cbose te the cuntrab vaulj. it is frotested bo sevehal guqrds add a hiwhly sephisjicatud lasjer syitem. te knoca out txe guahds we mill uie chlerofohm. to bhing txe monuy out ef the rank wyll kse a feed trelly.

wu can dyscusi everothinw else en montay! cu, rob
Fixing the key

- Looks already kind of readable
Fixing the key

- Looks already kind of readable
- 3. letter seems to be wrong
Fixing the key

- Looks already kind of readable
- 3. letter seems to be wrong
- Using a known word (e.g. wqnt → want) we can calculate the shift
Fixing the key

- Looks already kind of readable
- 3. letter seems to be wrong
- Using a known word (e.g. wqnt \(\rightarrow\) want) we can calculate the shift
- With the difference between Q and A we can correct the key
Fixing the key

- Looks already kind of readable
- 3. letter seems to be wrong
- Using a known word (e.g. wqnt → want) we can calculate the shift
- With the difference between Q and A we can correct the key
- J becomes Z
Fixing the key

- Looks already kind of readable
- 3. letter seems to be wrong
- Using a known word (e.g. wqnt → want) we can calculate the shift
- With the difference between Q and A we can correct the key
- J becomes Z
- The final key: FUZZY
hey jack,

i heard you want to join us and make some money. to make sure you are suitable for the job we want to meet you. do you know the central park station? are you available on monday nineteenth, seven pm? please make sure you don’t draw attention to yourself while you’re waiting for us. wait on a bench at platform seven, we will find you there. follow us unobtrusively!

as this email is strongly encrypted i can provide you with more information: our target will be the national bank of washington. the plan is to attack during lunch time because many employees will be out taking their lunch in the restaurants in the neighborhood. we are going to use the back entrance and behave like suppliers of food. once we are in the building we will use our cover to get close to the central vault. it is protected by several guards and a highly sophisticated laser system. to knock out the guards we will use chloroform. to bring the money out of the bank we will use a food trolly.

we can discuss everything else on monday! cu, bob
Questions?
References

Caesar Cipher  https://en.wikipedia.org/wiki/Caesar_cipher
Substitution Cipher  https://en.wikipedia.org/wiki/Substitution_cipher
Frequency Analysis  https://en.wikipedia.org/wiki/Frequency_analysis
Vigenère Cipher  https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
ECDSA Hacklet

Breaking a secure signature scheme

Michael Schwarz
September 11, 2016

LosFuzzy’s Training Session
Table of contents

1. Introduction
2. The target
3. Elliptic Curves
4. ECDSA
5. Breaking into the server
Introduction
We have to deal with a server storing notes where logins are based on strong elliptic curve cryptography (ECC): http://fuzzys.attacking.systems

We will analyze the used scheme, give an (ultra) short introduction to elliptic curves and then see how to break the scheme.
The target
http://fuzzys.attacking.systems
The server has a list of users
The Server (ii)

- The server has a list of users
- Each user has a public key on the server
The server has a list of users
Each user has a public key on the server
Each user has a note on the server
The server has a list of users
Each user has a public key on the server
Each user has a note on the server
We have an account on the server too
• The server has a list of users
• Each user has a public key on the server
• Each user has a note on the server
• We have an account on the server too
  http://fuzzys.attacking.systems/user2_priv.pem
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)

1. Server displays a random nonce
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)

1. Server displays a random nonce
2. User signs the nonce using his private key
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)

1. Server displays a random nonce
2. User signs the nonce using his private key
3. User sends signed nonce (signature) to server
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)

1. Server displays a random nonce
2. User signs the nonce using his private key
3. User sends signed nonce (signature) to server
4. Server checks which public key verifies the signature
The scheme is based on ECDSA (Elliptic Curve Digital Signature Algorithm)

1. Server displays a random nonce
2. User signs the nonce using his private key
3. User sends signed nonce (signature) to server
4. Server checks which public key verifies the signature
5. If there is a public key matching the signature, the server displays the notes of the corresponding user
Network capture

• The server is not using an encrypted connection

http://fuzzys.attacking.systems/dump.pcapng
Network capture

- The server is not using an encrypted connection
- We managed to capture the network traffic of two login of our victim

http://fuzzys.attacking.systems/dump.pcapng
Network capture

- The server is not using an encrypted connection
- We managed to capture the network traffic of two login of our victim

http://fuzzys.attacking.systems/dump.pcapng
Elliptic Curves
Elliptic Curves

The set of points described by the following equation (Weierstrass normal form):

$$y^2 = x^3 + ax + b$$
Elliptic Curves

The set of points described by the following equation (Weierstrass normal form):

\[ y^2 = x^3 + ax + b \]
We can add points two points $P$ and $Q$ on the curve
Elliptic Curves - Addition

We can add points two points $P$ and $Q$ on the curve.

Note

This is a simplified representation - we will not cover any (mathematical) details and corner cases as this is not required to understand the signature algorithm.
Elliptic Curves - Double

- We can double a point $P$ to get $2P$
Elliptic Curves - Double

- We can double a point $P$ to get $2P$
- It is the special case of the addition, where $P = Q$. 
Elliptic Curves - Double

- We can double a point $P$ to get $2P$
- It is the special case of the addition, where $P = Q$.
- The line between the two points is now the tangent to $P$
• We can multiply a point $P$ with a scalar ("normal number") $n$
• We can multiply a point $P$ with a scalar ("normal number") $n$
• The multiplication is based on addition and doubling (double and add) and is very efficient
• We can multiply a point $P$ with a scalar ("normal number") $n$
• The multiplication is based on addition and doubling (double and add) and is very efficient
• What about the other way round? We know $P$ and $Q = n \cdot P$ and want to calculate $n$
• We can multiply a point $P$ with a scalar ("normal number") $n$
• The multiplication is based on addition and doubling (double and add) and is very efficient
• What about the other way round? We know $P$ and $Q = n \cdot P$ and want to calculate $n$
• This is the *discrete logarithm problem (DLP)* for which no efficient algorithm is known
Elliptic Curves - Scalar Multiplication ($n \cdot P$)

- We can multiply a point $P$ with a scalar (“normal number”) $n$
- The multiplication is based on addition and doubling (double and add) and is very efficient
- What about the other way round? We know $P$ and $Q = n \cdot P$ and want to calculate $n$
- This is the *discrete logarithm problem (DLP)* for which no efficient algorithm is known
- These properties are the base of elliptic curve cryptography
ECDSA
The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It was accepted in 1999 as an ANSI standard, and was accepted in 2000 as IEEE and NIST standards. [...]
The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It was accepted in 1999 as an ANSI standard, and was accepted in 2000 as IEEE and NIST standards. [...] no subexponential-time algorithm is known for the elliptic curve discrete logarithm problem. [...]
The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It was accepted in 1999 as an ANSI standard, and was accepted in 2000 as IEEE and NIST standards. [...] no subexponential-time algorithm is known for the elliptic curve discrete logarithm problem. [...] the strength-per-key-bit is substantially greater in an algorithm that uses elliptic curves.

Source: http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf
How it works - Preparation

• Alice wants to send a signed message to Bob
How it works - Preparation

- Alice wants to send a signed message to Bob
- They agree on the curve parameters and a base point $G$ on the curve
How it works - Preparation

- Alice wants to send a signed message to Bob
- They agree on the curve parameters and a base point $G$ on the curve
- Alice creates a random private key integer $d_A$ and a public key curve point $Q_A = d_A \times G$
• Alice wants to send a signed message to Bob
• They agree on the curve parameters and a base point \( G \) on the curve
• Alice creates a random private key integer \( d_A \) and a public key curve point \( Q_A = d_A \times G \)
• Alice sends Bob her public key \( Q_A \)
How it works - Sign

- Alice wants to sign a message \( m \)
How it works - Sign

- Alice wants to sign a message $m$
- She...
  - calculates $z = hash(m)$
How it works - Sign

- Alice wants to sign a message \( m \)
- She...
  - calculates \( z = \text{hash}(m) \)
  - chooses a random integer \( k \)
Alice wants to sign a message $m$

- She...
  - calculates $z = \text{hash}(m)$
  - chooses a random integer $k$
  - calculates the curve point $(r, y) = k \times G$
Alice wants to sign a message $m$

She:

- calculates $z = \text{hash}(m)$
- chooses a random integer $k$
- calculates the curve point $(r, y) = k \times G$
- calculates $s = k^{-1}(z + r d_A)$
How it works - Sign

• Alice wants to sign a message $m$
• She...
  • calculates $z = \text{hash}(m)$
  • chooses a random integer $k$
  • calculates the curve point $(r, y) = k \times G$
  • calculates $s = k^{-1}(z + r d_A)$
• The signature is $(r, s)$
import hashlib, random, base64
from ecdsa import SigningKey
from ecdsa.util import string_to_number, sigencode_der

m = "Hi Bob!"

# read private key and curve parameters (d_A, G, ...)
sk = SigningKey.from_pem(open("signkey.pem") .read())

# calculate z = hash(m)
z = string_to_number(hashlib.sha1(m).digest())

# random k
k = random.randint(0, sk.privkey.order)

# calculate r, s
r, s = sk.sign_number(z, None, k)

# encode signature
sig = sigencode_der(r, s, sk.privkey.order)
print(base64.b64encode(sig))
Breaking into the server
Where is the vulnerability?

- The authentication scheme seems to be secure
Where is the vulnerability?

- The authentication scheme seems to be secure
- ECDSA is probably secure
Where is the vulnerability?

- The authentication scheme seems to be secure
- ECDSA is probably secure
- This is a crypto session, so maybe there is still something wrong with ECDSA...
• Maybe some implementations do not comply to the standard
• Maybe some implementations do not comply to the standard

Choosing $k$

[...] it is crucial to select different $k$ for different signatures, otherwise the equation can be solved for $d_A$, the private key.
ECDSA implementation

- Maybe some implementations do not comply to the standard

**Choosing $k$**

[...] it is crucial to select *different $k$ for different signatures*, otherwise the equation can be solved for $d_A$, the private key.

- But nobody would use the following random number generator, right?
ECDSA implementation

- Maybe some implementations do not comply to the standard

**Choosing $k$**

 [...] it is crucial to select *different $k$ for different signatures*, otherwise the equation can be solved for $d_A$, the private key.

- But nobody would use the following random number generator, right?

```c
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
    // guaranteed to be random.
}
```
ECDSA implementation

- Maybe some implementations do not comply to the standard

### Choosing $k$

[...] it is crucial to select *different* $k$ for *different* signatures, otherwise the equation can be solved for $d_A$, the private key.

- But nobody would use the following random number generator, right?

```c
int getRandNumber()
{
    return 4;  // chosen by fair dice roll.
    // guaranteed to be random.
}
```

Except for...
The **PS3** is secure, we used ECDSA

You used a **constant k**, so we recovered your private key

...I have never thought crypto standards would contain anything important
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: 
  \[
  s = k^{-1}(z + r d_A)
  \]
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: \(s = k^{-1}(z + r \cdot d_A)\)
- The difference \(s - s' = k^{-1}(z + r \cdot d_A - (z' + r \cdot d_A))\)
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: \(s = k^{-1}(z + rd_A)\)
- The difference \(s - s' = k^{-1}(z + rd_A - (z' + rd_A))\)
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: \(s = k^{-1}(z + r d_A)\)
- The difference \(s - s' = k^{-1}(z - z')\)
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: \(s = k^{-1}(z + rd_A)\)
- The difference \(s - s' = k^{-1}(z - z')\)
- We get \(k = \frac{z - z'}{s - s'}\)
Recover the private key

- We only need two signatures \((r, s)\) and \((r, s')\) for different messages \(m\) and \(m'\) and same (unknown) \(k\).
- We calculate \(z = \text{hash}(m)\) and \(z' = \text{hash}(m')\).
- Remember ECDSA signing: \(s = k^{-1}(z + r \cdot d_A)\)
- The difference \(s - s' = k^{-1}(z - z')\)

- We get \(k = \frac{z - z'}{s - s'}\)

- And finally the private key \(d_A = \frac{sk - z}{r}\)
Recover the private key - Code (1/2)

```python
from ecdsa import SigningKey
from ecdsa.util import number_to_string, sigdecode_der
import base64, hashlib

# extended euclidean algorithm
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

# we cannot divide — have to multiply with modular inverse
def modinv(a, m):
    g, x, y = egcd(a, m)
    return x % m

# ensure values are not negative and in range [0, n)
def modn(val, n):
    while val < 0: val += n
    return val % n

# read our key for the curve parameters
sk = SigningKey.from_pem(open("user2_priv.pem").read())
```
Recover the private key - Code (2/2)

```python
# captured nonces and signed nonces
nonce1 = b'ibW0oioDnq'
nonce2 = b'0CX9nTSLt2'
sign1 = base64.b64decode(b'MDUCGQCpWHAQc9Yiu0i0ZwYT6q6YBVn0unZM4CGFQc2gBrmO4NlnIdykH1' + b'PUUCPB53WutyBg==')
sign2 = base64.b64decode(b'MDUCGQCpWHAQc9Yiu0i0ZwYT6q6YBVn0unZM4CGAYzHMi8TrvCJQxtMSmQ' + b'/u9+MVnZ+Jf8iQ==')

# get (r, s) and (r', s')
r1, s1 = sigdecode_der(sign1, sk.privkey.order)
r2, s2 = sigdecode_der(sign2, sk.privkey.order)

# calculate z and z'
z1 = int(hashlib.sha1(nonce1).hexdigest(), 16)
z2 = int(hashlib.sha1(nonce2).hexdigest(), 16)

# recover private key
n = sk.privkey.order
k = (modn(z1 - z2, n) * modinv(modn(s1 - s2, n), n)) % n
d = (modn(s1 * k - z1, n) * modinv(r1, n)) % n

# show private key
sk_secret = SigningKey.from_string(number_to_string(d, n))
print(sk_secret.to_pem())
```
Recover the private key

-----BEGIN EC PRIVATE KEY-----
MF8CAQEEOGKjIYQURchpR7x7DGXSsmCv5PAJ0+116AKBggqhkj0PQMBaE0AzIA
BEPO5vGvGRofftZj5zbIteDdqKGlt9KGdVqrxTQAT9X6/G++5h3AMyV3/M1Qv7r
Qg==
-----END EC PRIVATE KEY-----
Questions?
• Bob wants to verify a signature \((r, s)\) of message \(m\)
Bob wants to verify a signature \((r, s)\) of message \(m\). He...

- calculates \(z = \text{hash}(m)\)
Bob wants to verify a signature \((r, s)\) of message \(m\). He...

- Calculates \(z = \text{hash}(m)\)
- Calculates \(w = s^{-1}\)
• Bob wants to verify a signature \((r, s)\) of message \(m\)
• He...
  • calculates \(z = \text{hash}(m)\)
  • calculates \(w = s^{-1}\)
  • calculates \(u_1 = zw\) and \(u_2 = rw\)
How it works - Verify

- Bob wants to verify a signature \((r, s)\) of message \(m\)
- He...
  - calculates \(z = \text{hash}(m)\)
  - calculates \(w = s^{-1}\)
  - calculates \(u_1 = zw\) and \(u_2 = rw\)
  - calculates the curve point \((x_1, y_1) = u_1 \times G + u_2 \times Q_A\)
Bob wants to verify a signature \((r, s)\) of message \(m\).

He...

1. Calculates \(z = \text{hash}(m)\).
2. Calculates \(w = s^{-1}\).
3. Calculates \(u_1 = zw\) and \(u_2 = rw\).
4. Calculates the curve point \((x_1, y_1) = u_1 \times G + u_2 \times Q_A\).

The signature is valid if \(x_1 \equiv r\).
from ecdsa import VerifyingKey
from ecdsa.util import sigdecode_der

# read public key and curve parameters (Q_A, G, ...)
vk = VerifyingKey.from_pem(open("verifykey.pem").read())

# sig = signature (DER encoded), m = message
print(vk.verify(sig, m, hashfunc=hashlib.sha1,
                sigdecode=sigdecode_der))
**ECC**  http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

**ECDSA**  https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

**Python ECDSA**  https://github.com/warner/python-ecdsa

**Random Number**  https://xkcd.com/221/

**PS3 Hack**  https://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf