

# Técnicas Criptográficas

Criptografia e Segurança da Informação (CSI)

Óscar Pereira

## Submissão de Trabalhos

Os trabalhos deverão ser realizados em **Python 3**, utilizando a biblioteca `cryptography`, se for caso disso. Os ficheiros a ser submetidos, incluindo o relatório, deverão ser colocados num arquivo `.zip`, e enviados por email para o docente. Adicionalmente, no **corpo desse email** deverão indicar o nome do curso, e dos elementos do grupo, juntamente com os respectivos números mecanográficos.

## Trabalho Prático I

Apresentam-se de seguida três criptogramas, cifrados com as cifras *affine*, de substituição, e Vigenère (não necessariamente por esta ordem). Efectue a respectiva criptanálise, e apresente um relatório indicando o texto limpo, e descrevendo todo o trabalho efectuado, incluindo o código. Não é suficiente apresentar apenas o texto limpo. O código deverá estar adequadamente documentado, e ser apresentado em ficheiros anexos, não incluso no relatório.

Nota: tratam-se de textos de língua inglesa. O criptograma contém a pontuação e espaçamento originais (i.e. só as letras são cifradas).

Criptograma I:

WMP BRHEMGQJ JD YSZ RWTBGQ ZA RZC QHAPMYO OJKHXEDA WLXCV QMMP DZTHWLG  
DGZMGJNYVJ XEJANX, FDX WCHS XYUWTZB YZ YQ LWQXWO CAYCZKH MT VTXZ  
DZECMUX. RKJJ ZHQTZTH EYCW PQCUD MYFJ RFLHS ZUJPYQ YCPC, WZR YSZ  
GNDOGQHEDTH NCYUFNOCUX WC JGZP XZ QONRCR, SVQ MLY LYD ULQO NUTEJRBUP.  
DY OFLX MYWJ OFHWP KXXE FDAP CANDOCG LO OJLNR F NARWP MI DKCFNPN RK  
RGOI XYWYWZ, FD KDSJ QKJPK, FYY VJGZPDQ BMDYD, LS ZSUTAZ DQZIC, LIB  
XPQCUFW CYJY ULYSDL LCZYW MMGWFTI. TYZ DZECMU MZJLJGZQ YSVR YSZPH  
QJPPJCGW JIDQWJO COJGZL BTGB XAZALJD MI DCCHU KCFZWDYU EJ JWPVR  
GCDRDNY! ZMPI ZJ WCDW DL RTIB YSVR GCDRDNY FDX IMZ YJR TYZ SJNPJLFC  
KDRXVJ, LIB KCVLFJ WSW QZU ITNRLSNO IWZH WMZNC TQ EHWXVLB, VLG DJ ZNEC  
KZYBYUD, QSFTI, JEX., MPR YSVR JLXF TQ RKJDZ NNYBBRRD NRXDZQVJD  
QHAPMYO AZAXQTVP GCZCGX JD HLOROJ, QKJPK, JEX., HZ PZDO DIXDR YSVR  
RLIW IZHCVYTX EWPZBV XPQW SVTH ZMGJNYVRHI DL JFMMSJ; DRW RFHSNZ  
RYSZPZNDZ FTFGB YSZW MLQC GPZL IPMGYJO? VT DR ND GQ TIBLF. CYJY GQ ECC  
HLNC TQ RKJ WPHJON RK OFH OJKHXEDA IZB WMCJSJMZPR YSZ ZTCGB, HCGFM D  
DIXDR FCZ GJDXCQIPY IWZH VJGZPDQ RGOI NNHTZQ, TO FFYIMW MZ GTFWRHI  
OFDY OFHWP FDX WCHS VL NXHCQXP YPTFIR TQ GQMPMGWJO TDWTVRLTY; ITC UKT  
RGOQ WCONPQC YSVR FYDKDQD AOTDZJB CZQHRMGGQL OFH TOYONLI JWPTFRZYY,  
YSZ EQZJBKTFIB, ECC GFGJ-IZB, UFB-GTR, RW WJHSSZGP DKYQNGP, JEX.-  
XZ

SQQTFC FWG ZNWX FFYDBDJ-ZTHW ZVLXEZB NY Y XEVRH ZA QFEPH? DR MLN  
RKEZL GPZL QZJQHJQ QDNO RKFE YOQ JSU CVAHX JD IZBQ MLQC GPZL UCJBXHPY  
ED OFH NMMVXTIE TQ Y KPR DGZMGJNYVJ XAZALJD; EZE ZB NMMVXTIE BP ADS  
JLOD BCW QJPPX DL XZHC IPBPHJ DLWJCHCGNLOC GPOUHJY RKJTM SFCZLWX; YQI  
DD BP YFHZPLW QJP TFM VJGZPDQ YMPJDOGF CVAHX WW YSDQ UCJAHXD, ZJ HSYV  
VBPNE RKJ AMURPM HCTNRHSNZ RK OFH XJQW PSRUJXZ ITCHQ, LN WMP GWFWDYQ  
RMCBMZPLG, WJRTOCMXSO, EZWG-GTR, HYN., NY RKJ RGOI NRDYP. PTCZMYJC,  
WMP NRXDDZLQTOW TQ KDPTIE ITNRLSNO UFNZQ GJ AUTDNGQL CYV MZCQ RMCYWT  
HCLBEHWLOG. HYQD XYVJD YUJ JL WPXMUI NFRBTIE YSVR F MYFJ HYB MZ  
PTODDLJO ZB ZXADXTJLDQ XPRXDZQ NQ YLIPY ED OFH NVPHKFG VJWZAWNZI RK  
OFH TIBLATYSDQD UKNNC SWPNCQY OFH OZQLWPY FMLMYFYPM; GFO WT JZWFTI D  
CVAH TIRHWXZBLFEZ EJERCHS OUR BPGWJ YGVYTIW CVAHX RMXQO ZH GZPB  
ODDINNPJW. NGU U. VJMMGJME CAUCZQVQJ CAUPMGPJYOCG HDRK ECGV ZWHHHE YQI  
AYLQPY. YSZ RKQNNUNYB IWZH WMP DLWDO FWZNQ GPOUHJY RZT KSUJ WPHJON LX  
OMOJCVZOD VLG DJKHVYHCV (VQ N CYYJ AMXSO ULYS NLLPJLV) LSLYP SQNQJPP  
TI FMLMYFYPM, FYY HAPMW YSDLJ DZCPX NGPUWZ HZSPEK; WSW HCCQ ECCVJ  
HMQLCZJV LMC HCJQVJO MQJ RGWM VLRYSZP KZM VJGZPDQ BCQJCVRLTYN, MLMBOD

OUR ZA WMPH DWP YONVZ, FYY WMPI WMP BLKQDAXQET RK OFH EVQN MZARRPN  
PFYDDHXE.

Criptograma II:

SJ SY VGJ JGG KUEH JG NACUSNA JHIJ QHIJ JHA QSYAYJ GL KIVOSVP, JHGVA  
QHG INA TAYJ AVJSJZAP JG JNUYJ JHASN GQV DUPWKAVJ, LSVP VAEAYYINM JG  
QINNIVJ JHASN NAZMSVW GV SJ, YHGUZP TA YUTKSJJAP JG TM JHIJ  
KSYEAZZIVAGUY EGZZAEJSGV GL I LAQ QSYA IVP KIVM LGGZSYH SVPSFSPUIZY,  
EIZZAP JHA RUTZSE. JHA KGYJ SVJGZANIVJ GL EHUNEHAY, JHA NGKIV EIJHGZSE  
EHUNEH, AFAV IJ JHA EIVGVSIIJSGV GL I YISVJ, IPKSJY, IVP ZSYJAVY  
RIJSAVJZM JG, I "PAFSZ'Y IPFGEIJA." JHA HGZSAYJ GL KAV, SJ IRRAINY,  
EIVVGJ TA IPKSJJAP JG RGYJHUKGUY HGVGUNY, UVJSZ IZZ JHIJ JHA PAFSZ  
EGUZP YIM IWISVYJ HSK SY OVGQV IVP QASWHAP. SL AFAV JHA VAQJGVSIV  
RHSZGYGRHM QANA VGJ RANKSJJAP JG TA CUAYJSGVAP, KIVOSVP EGUZP VGJ LAAZ  
IY EGKRZAJA IYYUNIVEA GL SJY JNUJH IY JHAM VGQ PG. JHA TAZSALY QHSEH  
QA HIFA KGYJ QINNIVJ LGN, HIFA VG YILAWUINP JG NAYJ GV, TUJ I YJIVPSVW  
SVFSJIJSGV JG JHA QHGZA QGNZP JG RNGFA JHAK UVLGUVVAP. SL JHA  
EHIZZAVWA SY VGJ IEEARJAP, GN SY IEEARJAP IVP JHA IJJAKRJ LISZY, QA  
INA LIN AVGUWH LNGK EANJISVJM YJSZZ; TUJ QA HIFA PGVA JHA TAYJ JHIJ  
JHA ABSYJSVW YJIJA GL HUKIV NAIYGV IPKSJY GL; QA HIFA VAWZAEJAP  
VGJHSVW JHIJ EGUZP WSFA JHA JNUJH I EHIVEA GL NAIHESVW UY: SL JHA  
ZSYJY INA OARJ GRAV, QA KIM HGRA JHIJ SL JHANA TA I TAJJAN JNUJH, SJ  
QSZZ TA LGUVP QHAV JHA HUKIV KSVP SY EIRITZA GL NAEASFVW SJ; IVP SV  
JHA KAIVJSKA QA KIM NAZM GV HIFSVW IJJISVAP YUEH IRRNGIEH JG JNUJH, IY  
SY RGYSTZA SV GUN GQV PIM. JHSY SY JHA IKGUVJ GL EANJISVJM IJJISVITZA  
TM I LIZZSTZA TASVW, IVP JHSY JHA YGZA QIM GL IJJISVSW SJ.

Criptograma III:

HG UOVI UJXXUTV VI GI BX, NRG H UB CIG USUTX  
GQUG UCA MIBBRCHGA QUV U THKQG GI ZITMX UCIGQXT GI NX MHFHOHVXL. VI  
OICK UV GQX VRZZXTXTV NA GQX NUL OUS LI CIG HCFIPX UVVHVGUCMX ZTIB  
IGQXT MIBBRCHGHXV, H MUCCIG ULBHG GQUG JXTVICV XCGHTXOA RCMICXMGXL  
SHGQ GQXB IRKQG GI VGXJ HC UCL TXYRHTX GQUG U MICLHGHC IZ GQHCKV SHGQ  
SQHMV UOO SQI UTX LHTXMGOA HCGXTXVGXL UJXXUT GI NX VUGHVZHL, VQIROL  
NX JRG UC XCL GI NXMURVX HG HV U VMUCLUO GI JXTVICV VIBX GQIRVUCLV IZ  
BHOXV LHVUGC, SQI QUFV CI JUTG IT MICMXTC HC HG. OXG GQXB VXCL  
BHVHICUTHXV, HZ GQXA JOXUVX, GI JTXUMQ UKUHCVG HG; UCL OXG GQXB, NA  
UCA ZUHT BXUCV (IZ SQHMV VHOXCMHCK GQX GXUMQXTV HV CIG ICX), IJJIVX  
GQX JTIKTXVV IZ VHBHOUT LIMGTHCXV UBICK GQXHT ISC JXIJOX. HZ  
MHFHOHVUGHC QUV KIG GQX NXGGXT IZ NUTNUTHVB SQXC NUTNUTHVB QUL GQX  
SITOL GI HGVXOZ, HG HV GII BRMQ GI JTIZXVV GI NX UZTUHL OXVG  
NUTNUTHVB, UZGXT QUFHCK NXXC ZUHTOA KIG RCLXT, VQIROL TXFHFV UCL  
MICYRXT MHFHOHVUGHC. U MHFHOHVUGHC GQUG MUC GQRV VRMMRBN GI HGV  
FUCYRHVQXL XCXBA, BRVG ZHTVG QUFV NXMIBX VI LXXXCXTUGX, GQUG CXHGQXT  
HGV UJJIHCGXL JTHXVGV UCL GXUMQXTV, CIT UCANILA XOVX, QUV GQX

MUJUMHGA, IT SHOO GUPX GQX GTIRNOX, GI VGUCL RJ ZIT HG. HZ GQHV NX VI,  
GQX VIICXT VRMQ U MHFHOHVUGHIC TXMXHFXV CIGHMX GI YRHG, GQX NXGGXT. HG  
MUC ICOA KI IC ZTIB NUL GI SITVX, RCGHO LXVGTIAXL UCL TXKXCXTUGXL  
(OHPX GQX SXVGXTC XBJHTX) NA XCXTKXGHM NUTNUTHUCV.

## Trabalho Prático II

Pretende-se analisar os 20 criptogramas disponibilizados no ficheiro <https://randomwalk.eu/public/tp2-ciphertexts.txt>. Foram gerados utilizando a *One-Time Pad* de uma forma incorrecta, nomeadamente porque se utilizou uma sequência pseudo-aleatória, gerada por um gerador aleatório fraco (o da linguagem Python). Adicionalmente, sabe-se que os criptogramas correspondem a duas mensagens de texto limpo, cifradas com chaves diferentes—excepto no caso de dois criptogramas, que foram cifrados utilizando a mesma chave.

Analise os criptogramas fornecidos, com vista a descobrir quais são esses dois criptogramas. Descreva o trabalho efectuado num curto relatório (máx. 2 páginas). Em ficheiros anexos, deverá também ser fornecido o código utilizado, adequadamente comentado.

**Nota:** a cifra OTP aqui aplicada **não corresponde a um bitwise-XOR**. Se assim fosse, no criptograma iriam existir bytes que não corresponderiam a caracteres na gama *A – Z*—e portanto teriam que ser representados com notação hexadecimal. O que não é o caso, porque se utilizou a OTP aplicada ao alfabeto A-Z. Assim a operação utilizada para cifrar não foi o XOR, mas sim a soma módulo 26. A respeito da decifragem, assina-se que módulo 26, ao contrário do que acontece módulo 2 (XOR), **soma a subtracção são operações diferentes**.

## Trabalho Prático III

Pretende-se ilustrar como mesmo utilizando uma cifra por blocos segura, a segurança pode ser gravemente quebrada, se essa cifra for utilizada em conjunto com um modo inseguro. Para esse efeito, obtenha uma imagem à sua escolha, de preferência com um fundo uniforme (i.e. da mesma cor), e cifre-a utilizando o AES no modo ECB.

Para tal, pode utilizar a biblioteca Python `cryptography`, disponível em <https://cryptography.io/>. De seguida, é necessário que altere o ficheiro do criptograma, de modo a ser possível visualizá-lo como imagem. Uma sugestão é, se utilizar imagens no formato *bitmap* (extensão `.bmp`), alterar o header do criptograma da seguinte forma:

```
$ dd if=original.bmp of=ciphertext.bmp bs=1 count=54 conv=notrunc
```

De seguida, repita o processo com a mesma cifra, mas agora utilizando um modo seguro, como o CBC ou CTR. Modifique também este criptograma de modo a ser possível visualizá-lo como imagem. Discuta as mudanças que se observam entre ambos criptogramas, à luz do que aprendeu sobre modos de utilização de cifras por blocos, e da respectiva segurança (ou falta dela).

Apresente os criptogramas obtidos, bem como a imagem original, e a discussão subsequente, num curto relatório (máx. 2 páginas). Deverá também apresentar o código utilizado, adequadamente comentado, em ficheiros anexos.

## Trabalho Prático IV

**Primeira parte.** Pretende-se utilizar a biblioteca `cryptography` para implementar os três esquemas estudados para utilizar, na mesma construção, primitivas que fornecem integridade com primitivas que fornecem confidencialidade. No caso deste exercício, a primeira será o HMAC, e a segunda a cifra ChaCha20. Note-se que tratando-se de uma cifra sequencial, não é necessário qualquer padding. Os três esquemas são os seguintes:

- `encrypt-then-mac`
- `encrypt-and-mac`
- `mac-then-encrypt`

Para isso deverá completar as funções `etm()`, `eam()`, e `mte()`, nos ficheiros `enc.py` e `dec.py`, disponibilizados em [http://randomwalk.eu/media/Teaching/TC/Ficheiro\\_TP4.zip](http://randomwalk.eu/media/Teaching/TC/Ficheiro_TP4.zip) de forma incompleta. O objectivo é que o programa `enc.py` escreva o criptograma e/ou tag para um ficheiro, e o programa em `dec.py` leia desse ficheiro a informação necessária para decifrar o criptograma e verificar a integridade. No fim, deverá ser mostrada (`print'd`) a mensagem decifrada. Notas:

- Não deverá alterar a mensagem de texto limpo que se encontra no ficheiro `enc.py`, nem os nomes dos ficheiros `.dat`.
- Se tiver que armazenar informação adicional (e.g. nonce), poderá adicionar campos que contenham o comprimento das diferentes partes da mensagem. Assinala-se, no entanto, que neste trabalho em concreto, *existe uma estratégia que dispensa essa complicação extra*.
- **Esta parte do trabalho dispensa relatório.** Contudo, deverá documentar adequadamente o código produzido.

**Segunda parte.** Considere o CBCMAC, utilizando a cifra AES, para mensagens de tamanho fixo, igual a dois blocos do AES. Podemos enfraquecer este MAC, de dois modos distintos:

- Utilizando um IV aleatório, ao invés de um valor fixo (tipicamente uma string de zeros).
- Utilizando como *tag* todos os blocos do criptograma, em vez de apenas o último bloco.

Elabore um curto relatório (máx. 2 páginas) descrevendo como produziria uma falsificação para cada um deles. Adicionalmente, deverá implementar **apenas um** dos ataques, completando as funções `cbcmac(...)`, `verify(...)` e `produce_forger(...)` no ficheiro `forger_stub.py`, disponibilizado no mesmo zip indicado acima. Os comentários existentes no ficheiro fornecem informações adicionais.

*Dica: dizer que o IV é aleatório significa que é lícito o atacante submeter uma falsificação com um IV à sua escolha—o que permite manipular um dos blocos da mensagem.*

## Trabalho Prático V

Neste trabalho não é necessário utilizar a biblioteca *cryptography*.

**Primeira parte.** Resolva os seguintes sistemas de congruências modulares.

$$\text{a) } \begin{cases} x \equiv 48 \pmod{13} \\ x \equiv 57 \pmod{23} \\ x \equiv 39 \pmod{27} \end{cases} \quad \text{b) } \begin{cases} 19x \equiv 21 \pmod{16} \\ 37x \equiv 100 \pmod{15} \end{cases}$$

Não é necessário indicar todos os cálculos efectuados, mas é necessário indicar **todos os passos** que foram seguidos—acrescidos das justificações que achar necessárias.

**Nota:** existem programas que resolvem automaticamente este tipo de congruências. Por razões óbvias, essa abordagem não é permitida...

**Segunda parte.** O criptograma que se apresenta em baixo foi obtido através da utilização “textbook” do RSA. Ou seja, o texto limpo, previamente extirpado de sinais de pontuação, e capitalizado, foi codificado para inteiros como se indica a seguir, e depois cada inteiro foi cifrado utilizando a cifra RSA, de modo determinístico, com os seguintes parâmetros:  $e = 17$ ,  $n = 213271$ . O esquema de codificação utilizado foi o seguinte: as letras A, ..., Z foram mapeadas para  $0, \dots, 25$ , e o caractere espaço para 26. De seguida, considerando este inteiro um elemento de  $\mathbb{Z}_{27}$ , cada conjunto de três letras (ou espaços) seguidos foi mapeado num inteiro segundo o seguinte polinómio:

$$27^2 L_1 + 27 L_2 + L_3 \tag{1}$$

Foi ao inteiro resultante desta transformação que foi aplicada a cifragem RSA conforme descrito. Utilize o facto de o módulo ser pequeno (e portanto, relativamente fácil de factorizar) para quebrar esta instância de utilização do RSA. Pode, se entender necessário, utilizar uma implementação já existente do algoritmo de Euclides, mas de resto *deverá implementar todo o código que necessitar*.

Deverá submeter o código, devidamente comentado, acompanhado de um relatório que apresente o texto limpo, e explique a abordagem seguida.

6876, 90542, 209524, 180723, 68349, 24407, 1927, 183075, 37458,  
77446, 197372, 14551, 148450, 213237, 55592, 56745, 15085, 103645,  
154406, 67322, 2002, 39417, 127400, 178722, 76999, 37458, 79735,  
198950, 161111, 69856, 142050, 22632, 39091, 16950, 168529, 162080,  
83943, 72950, 24407, 207238, 18354, 38021, 186689, 59975, 125376,  
161647, 195221, 44657, 48754, 96701, 72273, 108266, 209524, 16077,  
112276, 69856, 142050, 22632, 84465, 162080, 36730, 27249, 34758,  
79735, 200474, 186981, 99905, 81699, 56760, 56967, 151769, 67608,  
137974, 76557, 187031, 103901, 128885, 148040, 128883, 54852,  
166919, 168279, 44550, 19456, 80788, 141636, 159372, 90688, 35758,  
168747, 142924, 190769, 174948, 2791, 69856, 142050, 22632, 84465,  
162080, 157426, 59221, 65034, 158258, 128733, 108251, 11016, 3376,  
31144, 79735, 162990, 200008, 141687, 136850, 22342, 196127, 117300,  
100284, 64381, 36124, 93455, 97454, 158631, 60424, 91786, 209412,  
57924, 183075, 101801, 55880, 56760, 68019, 164064, 2791, 37458,



209662, 188390, 68954, 169696, 168434, 115729, 156200, 52926,  
73555, 193991, 37458, 12591, 64130, 61216, 79735, 132216, 194613,  
167517, 196127, 84228, 57242, 122520, 123552, 103901, 176508,  
43547, 145243, 69650, 209524, 202257, 99142, 51498, 162203, 117210,  
127989, 102955, 77762, 24166, 147550

**Nota:** o esquema RSA utilizado foi o apresentado na aula, *que não faz uso da função  $\phi$  de Euler.*