KSI

Korespondenční Seminář z Informatiky

Milé řešitelky a milí řešitelé,

po vánočním a novoročním zimním spánku se Vám hlásíme s další sadou KSI. Čtvrtá sada je výzva - jde o kryptosadu. V obálce, která Vám přišla poštou byste měli mít tři papíry, které jsou v případě potřeby dohledatelné i na webu KSI:

1. řešení předchozí sady,
2. zadání příkladů, které právě čtete,
3. zadání šifry z příkladu 2 na zvláštním papíře.

Teď už k samotnému tématu. Šifrování a kryptografie jsou způsoby, jak docílit toho, aby Váš text četl jen ten, kdo je povolaný. V informatice je velice často algoritmus, kterým jsou data zašifrovaná, známý, ale ten, kdo nezná klíč, je (pokud je kryptosystém dobrý) nucen trávit výpočtem a zkoušením mnoha možností příliš dlouhou dobu, a nebo počkat než budou vyvinuty rychlejší počítače. Mezitím zpráva třeba přestane být aktuální a nebo přestane být tajná. V rozumně krátkém čase si dokáže zprávu přečíst jen ten, kdo má tajnou informaci - klíč.

Pokud jde o šifrovací táborovou hru, nepředpokládá se, že by někoho bavilo louskat nuly a jedničky na počítači a povolaným je ten, kdo má správný nápad (*vždyť je to morseovka! je to jenom napsané zrcadlově! ...*). V této sadě ochutnáte od obou druhů šifer.

Základy šifrování a pojmy

Úlohou šifrování je změna textu tak, aby ho nemohl přečíst nikdo kromě toho, kdo zná tajný klíč. Zadání je tedy jasné a triviální, stojí však za to podívat se na něj blíže. Každá šifra se skládá ze dvou částí: šifrovací funkce a dešifrovací funkce. Pro jasnější zápis zavedeme následující konvenci: šifrovací funkci budeme označovat písmenem E[1] a dešifrovací funkci budeme označovat písmenem D[2]. První zajímavá věc je, že tyto funkce nemusí vypadat stejně, naopak, můžou být velmi odlišné. Přitom však musí splňovat přirozenou podmínku

$$\mathrm{D}(\text{klíč}, \mathrm{E}(\text{klíč}, \text{zpráva})) = \text{zpráva}$$

tedy že po dešifrování zašifrované zprávy se dostaneme k původní zprávě.

Kryptografové stojí před úlohou navrhnout dvě funkce E a D. Možná trochu překvapivé je, že v zásadě existují jenom dvě možnosti, jak transformovat vstup:

Substituce – nahrazování písmen abecedy jinými

Transpozice – změna pořadí písmen ve zprávě

Obě tyto techniky byly využité už dávno při počátcích kryptografie - asi nejznámější šifrou využívající substituce je Caesarova šifra a jednou z prvních transpozičních šifer je spartská Skytalé. Kombinace těchto dvou přístupů je samozřejmě možná - nic nebrání po nahrazení písmen zprávy je ještě popřehazovat. Po troše teorie se teď pojďme podívat na příklady několika šifer.

Několik historických šifer

Začněme několika historickými šiframi. Ve všech počítáme s abecedou o 26 písmenech

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

[1] Z anglického „encrypt“

[2] Z anglického „decrypt“

Caesarova šifra Tuto šifru (resp. jednu z jejích variací) používal pro vojenskou komunikaci Julius Caesar a popsal ji v Zápiscích o válce galské. Jde jednoduše o to, že text, který chceme zapsat, posuneme v abecedě o k písmen dopředu nebo dozadu (všechny písmena o to stejné k a stejným směrem). Například posun textu *ahoj želvo* o 4:

$$AHOJZELVO \rightarrow ELSNDIPZS$$

Jedná se o substituční šifru (každé písmeno abecedy je nahrazeno pevně daným znakem v celé šifře).

Albertiho šifra Vymyslel ji v 15. století Leon Battista Alberti. Oproti Caesarově šifře používá dva posuny. Lichá písmena původního textu posuneme v abecedě o n_1, sudá o n_2. Příklad pro $n_1 = 25, n_2 = 2$:

$$AHOJZELVO \rightarrow ZJNLYGKWN$$

Jak uvidíme dále, je to vlastně speciální případ Vigenèrovy šifry s dvoupísmenným klíčem.

Vigenèrova šifra Albertiho šifru vylepšil v 16. století Giovan Battista Bellaso. Ve své době byla tato šifra považována za nerozluštitelnou. Klíčem k Vigenèrově šifře je slovo, pomocí kterého byla zpráva zašifrovaná. Například zvolme klíč *krysa* a zkusme pomocí tohoto klíče zašifrovat text *nejkulatoulinkatější zpráva*. Nejprve zkopírujeme klíč pod původní text několikrát za sebou (aby vystačil na celou zprávu). Potom posuneme každé písmeno zprávy v abecedě o tolik, kolikáté je v abecedě písmeno z klíče pod ním. Poslední řádek je zašifrovaný text.

```
NEJKULATOULINKATEJSIZPRAVA  ... původní zpráva
          +
KRYSAKRYSAKRYSAKRYSAKRYSAK  ... opakovaný klíč
          =
XVHCUVRRGUVZLCADVHKIJGPSVK  ... kryptotext
```

K ručnímu šifrování a dešifrování je pěknou pomůckou tzv. Vigenèrův čtverec. Jde o čtverec 26×26 písmen, ve kterém je na prvním řádku abeceda, na druhém abeceda posunutá o 1 (t.j. BCDEF...XYZA), na třetím posunutá o 2, až na 26. posunutá o 25. Například k posunu písmene N o pořadové číslo písmene K stačí najít ve čtverci písmeno v průsečíku řádku začínajícího N a sloupce začínajícího K. Je to X.

Vernamova šifra Na cestě historií přeskočíme do 19. století k pravděpodobně nejznámější a také nejpoužívanější šifře, kterou je Vernamova šifra, též známá jako One-time pad. Její podstatou je kódování pomocí úplně náhodného klíče, který je stejně dlouhý jako posílaná zpráva a který je použitelný pouze jednou. Výhoda tohoto kryptosystému je obrovská - nejen, že je matematicky dokazatelně neprolomitelný, ale navíc je i optimální[3]! Jeho nevýhoda je však stejně velká: protože klíč je stejně dlouhý jako zpráva a můžeme ho použít pouze jednou, tak jsme si příliš nepomohli - místo bezpečného přenosu zprávy musíme řešit bezpečný přenos klíče.

Tato šifra vznikla původně v textové variantě[4], dnes se používá v binární podobě s použitím operace XOR na bitech [5]:

$$C = M \oplus K$$

Při jejím použití nesmíme zapomenout na zásadu: každý klíč použít pouze jednou! A proč vlastně? Předpokládejme, že dvě stejně dlouhé zprávy M_1 a M_2 byly zašifrované stejným náhodným klíčem:

[3]Vzhledem k délce klíče - neexistuje kryptosystém, který by byl neprolomitelný a používal kratší klíč než Vernamova šifra.

[4]http://cs.wikipedia.org/wiki/Vernamova_šifra

[5]Opět přijmeme běžnou konvenci a původní zprávu budeme označovat písmenem M (z anglického "message"), zašifrovanou zprávu písmenem C (z anglického "ciphertext") a tajný klíč písmenem K

$$C_1 = M_1 \oplus K,\ C_2 = M_2 \oplus K$$

V tom případě může útočník provést XOR obou zašifrovaných zpráv:

$$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2 \oplus K \oplus K = M_1 \oplus M_2$$

Všimněme si, že $M_1 \oplus M_2$ neobsahuje žádnou náhodnost - jde o XOR dvou zpráv přirozeného jazyka. Pomocí statistické analýzy už tedy není žádný velký problém z $M_1 \oplus M_2$ najít původní zprávy M_1 a M_2. Tento útok je známý jako Two-time pad.

Magický XOR Při studiu kryptografie bude XOR pravděpodobně funkce, na kterou narazíte nejčastěji. XOR můžeme definovat následující tabulkou:

		$\oplus$
1	1	0
1	0	1
0	1	1
0	0	0

V předcházejícím úseku jsme při útoku na two-time pad využili několik vlastnosti XORu. Z tabulky vidíme, že je komutativní, takže můžeme změnit pořadí provádění operací XOR:

$$(M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2 \oplus K \oplus K$$

Všimněme si, že XOR bitu sám se sebou dá vždy výsledek 0, proto[6]

$$K \oplus K = 0$$

Další pěkná vlastnost je, že XOR libovolného bitu s 0 je právě tento bit, takže

$$M_1 \oplus M_2 \oplus 0 = M_1 \oplus M_2$$

Skutečná hodnota XORu pro kryptografii však leží v matematickém tvrzení: výsledkem XORu libovolné posloupnosti bitů s náhodnou posloupností bitů je opět náhodná posloupnost bitů. Představme si na místě libovolné posloupnosti bitů naši zprávu a náhodnou posloupnost bitů jako náš tajný klíč - nejde o nic jiného než nám známou Vernamovu šifru. Uvedené tvrzení pak říká, že když si dáme dobrý pozor a tajný klíč se bude pro útočníka jevit jako náhodný (nejen, že bude vygenerovaný náhodně, ale útočník o něm nesmí mít žádné další informace), potom výsledek bude také úplně náhodný. A v tom spočívá dokonalá bezpečnost Vernamovy šifry!

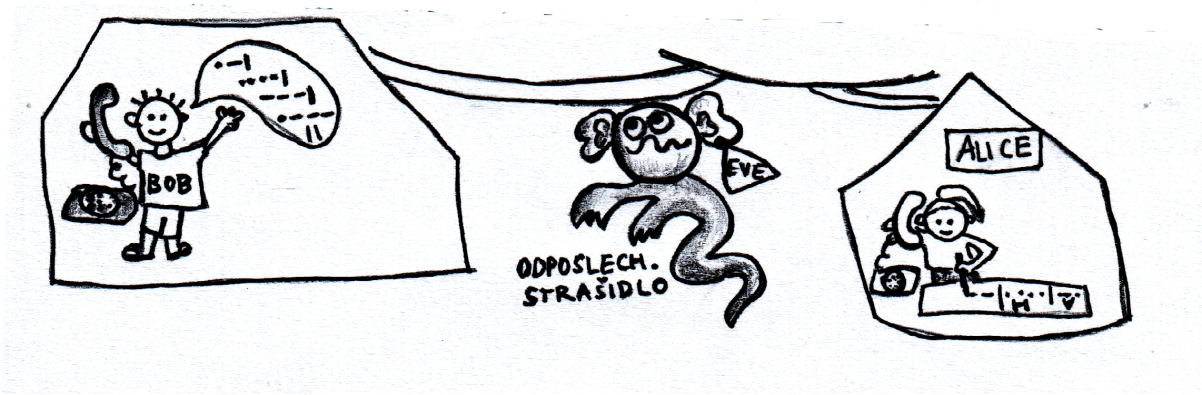
Kryptografie, jak ji známe dnes

Moderní kryptografie je už poměrně vzdálená těmto historickým šifrám. Dnes už nepotřebujeme šifrovat jen text, ale také obrázky, video, zvuk či jiná binární data. Vstupem i výstupem šifer jsou tedy posloupnosti jedniček a nul[7] místo písmen a číslic. Podle způsobu zpracování zprávy můžeme dnešní kryptosystémy rozdělit na dva druhy: blokové a proudové.

Proudové šifry se vyznačují tím, že zpracovávají data bit po bitu. Implementací se podobají Vernamově šifře - data jsou XORovaná s náhodnými bity - přesněji pseudonáhodnými bity, které jsou generované z tajného klíče. Příkladem je šifra RC4 používaná v protokolu WEP na šifrování WiFi.

[6] Při použití XOR na posloupnost bitů ho aplikujeme bit po bitu, takže výsledek není ve skutečnosti 0, ale posloupnost nul.

[7] Pro přehlednost používáme hexadecimální zápis. V něm jsou byty chápané jako čísla od 0 do 255 a zapsané v hexadecimální soustavě. Např. 0 = 00, 1 = 01, ..., 255 = FF.



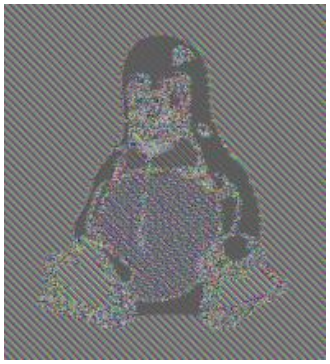
Blokové šifry naopak zpracovávají data po blocích předem stanovené délky. Všimněme si, že při tomto způsobu zpracování můžeme využít transpozici a bity v každém bloku popřehazovat. Blokové šifry bývají sice bezpečnější, na druhou stranu ale také pomalejší. Důležitým parametrem je délka bloku - čím je blok delší, tím je šifra bezpečnější (lze totiž popřehazovat víc bitů a měli bychom použít delší klíč). Při použití blokových šifer však musíme řešit dva zásadní problémy:

1. Očekáváme, že data budou mít délku dělitelnou délkou bloku, což se ale stane jen zřídkakdy.
2. Bloky jsou šifrované nezávisle, takže pokud zpráva obsahuje dva úplně stejné bloky, tak i v šifrované podobě budou tyto bloky shodné.

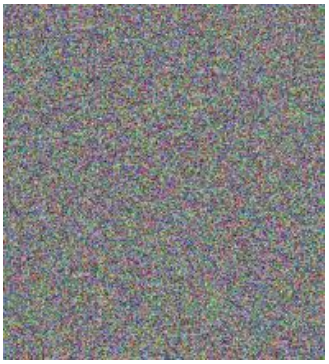
První problém se řeší pomocí takzvaného paddingu, tedy doplnění zprávy na správnou délku tak, aby bylo možné po dešifrování padding rozpoznat od zprávy samotné a zahodit ho. Nejjednodušší padding je přidání binárního řetězce `10...0` na konec zprávy, kde počet nul závisí na počtu bitů, které chybí na zarovnání délky bloku. Odstranění paddingu je jednoduché: po dešifrování zprávy zahodíme všechny nuly od konce až po první jedničku. Přidání paddingu ke zprávě je trochu složitější. Problematický je paradoxně ten případ, kdy je délka bloku původní délky dělitelná délkou bloku - tedy když padding vůbec nepotřebujeme. Pokud končí původní zpráva několika nulami, tak po přijetí a dešifrování budou zahozené, protože příjemce se nemá jak dovědět, že byly součástí zprávy a ne paddingu. V takovém případě tedy musíme na konec zprávy přidat celý extra blok navíc, tvořený řetězcem `10...0`.

Druhý bod se na první pohled ani nemusí jevit jako problém - co na tom, že útočník vidí, které bloky původní zprávy jsou stejné, když stejně neví jejich obsah? Za vše hovoří obrázek[8]:

Původní obrázek



Šifrovaný blokovou šifrou



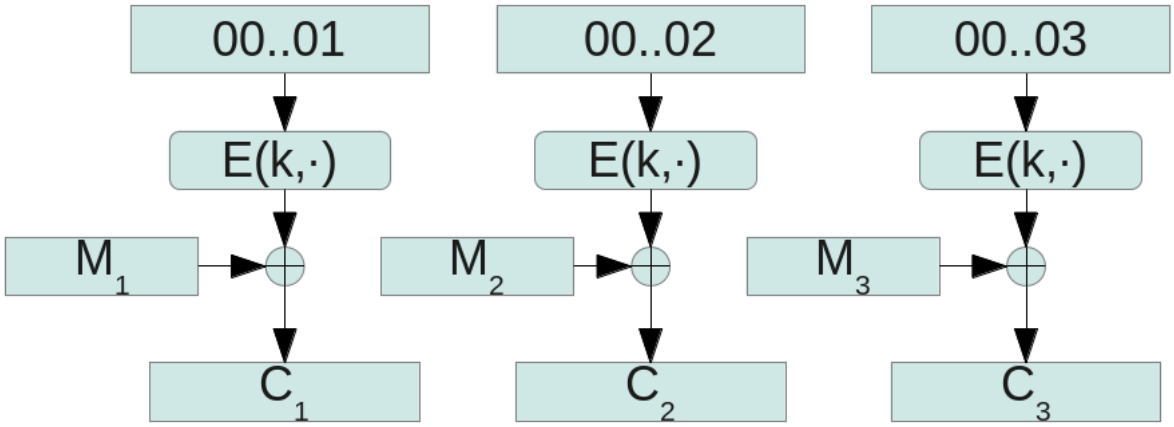
Šifrovaný bezpečným způsobem (CBC, CTR, ...)

Vidíme, že nakonec je přece jen žádoucí, aby dva stejné bloky na různých místech zprávy byly šifrované různě. Způsob, který toho docílíme bez úpravy blokové šifry samotné, se běžně nazývá mód šifrování.

[8]zdroj: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation#Electronic_codebook_.28ECB.29

Counter mód (CTR) Jak od sebe nejjednodušeji odlišit bloky zprávy? Očíslovat je. Jak ale použít čísla bloků tak, aby stejné bloky měly různé zašifrované podoby?

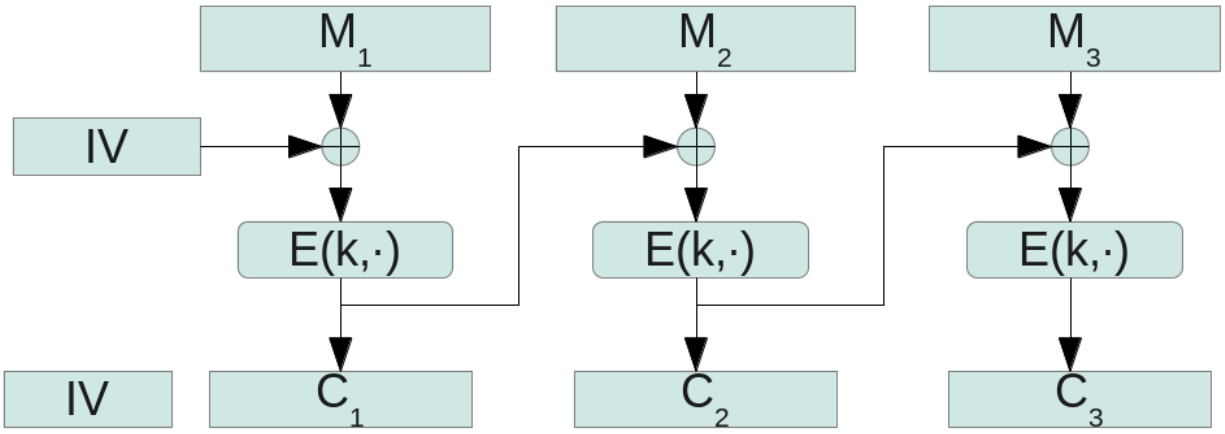
První nápad by mohl být jednoduché XORování zašifrovaných bloků číslem bloku. Tento přístup je však nesprávný - útočník přece ví, že první zpráva byla XORovaná číslem 1, druhá číslem 2 atd., takže dokáže číslování odstranit (vrátit se k podobě zašifrované zprávy před XORováním s číslem bloku). Vidíme, že číslo bloku musí být také zašifrované. Zkusme to tedy opačně: předpokládejme, že budeme pomocí blokové šifry šifrovat čísla bloků a výsledek budeme XORovat s bloky původní zprávy. Zde je schéma celého procesu takového šifrování:



tedy $C_1 = M_1 \oplus E(K, 1)$, $C_2 = M_2 \oplus E(K, 2)$ ...

Tento způsob šifrování má několik výhod. Protože známe klíč a čísla bloků předem, je možné vypočítat potřebné hodnoty dopředu, takže dešifrování zprávy může být o to rychlejší. Také je velmi jednoduché tento mód šifrování paralelizovat. A nakonec, všimněte si jeho jednoduchosti: dešifrování je (překvapivě?) stejné jako šifrování (stačí vyměnit M a C)!

Cipher Block Chaining (CBC) Další zdroj proměnlivosti (kromě pořadí bloků) můžeme hledat ve zprávě samotné. Předpokládejme, že máme N po sobě jdoucích shodných bloků. Jak můžeme docílit, aby každý z nich byl zašifrovaný jinak? Můžeme je zkusit XORovat s nějakým náhodným řetězcem předtím, než je pošleme na zašifrování blokové šifře. Kde najdeme takovýto řetězce? Jednoduše: využijme předchozí zašifrovaný blok! Opět - za vše hovoří obrázek:

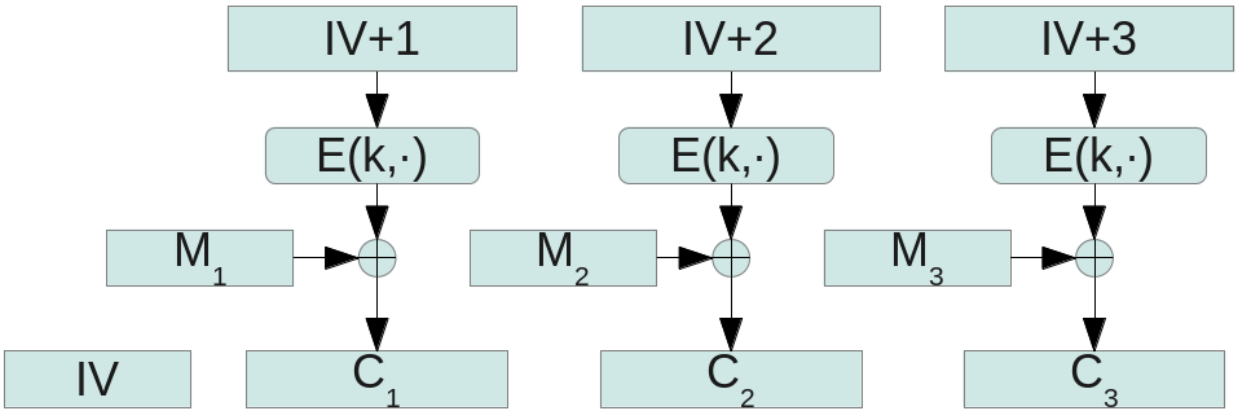


Všimněme si problému s prvním blokem: abychom ho měli s čím XORovat, přidáme na začátek zprávy náhodně vygenerovaný bitový řetězec - Inicializační vektor (IV). Musíme ho však poslat v otevřené, nezašifrované podobě připojený k naší zprávě (jinak by příjemce zprávy nemohl provést dešifrování).

Randomizovaný CTR Podívejme se ještě jednou na CTR mód. Možná to na první pohled není úplně zřejmé, ale šifrování předem známých čísel může představovat vážné riziko. Všimněme

se, že bloková šifra v CTR módě se chová jako Vernamova šifra, přičemž zprávu neXORujeme s náhodným klíčem, ale s jakousi pseudonáhodnou posloupností vygenerovanou postupným šifrováním čísel od 1. Předpokládejme, že na šifrování dvou zpráv použijeme stejný klíč. Tak jako při Vernamově šifře by útočník mohl využít útok na Two-time pad.

Od řešení nejsme daleko: inspirujeme se módem CBC a na začátek zprávy přidáme inicializační vektor - náhodné číslo. Při šifrování každého následujícího bloku tedy jednoduše zvýšíme IV o 1. Tím docílíme, že žádné dva stejné bloky zprávy nebudou stejně zašifrované, a zároveň zabezpečíme, že dvě různé zprávy budou začínat číslování bloků od různého čísla (tím se vyhneme two-time pad).



Šifrovaní: $C_1 = M_1 \oplus E(K, IV + 1)$, $C_2 = M_2 \oplus E(K, IV + 2)$, ...
Dešifrovaní: $M_1 = C_1 \oplus E(K, IV + 1)$, $M_2 = C_2 \oplus E(K, IV + 2)$, ...

Pokud jste pozorně četli, jistě pro Vás nebude problém navrhnout mód šifrování, který z libovolné blokové šifry udělá šifru proudovou.

Jak prolomit bezpečnou šifru?

Možná se Vám zdá, že padding a módy šifrování jsou spíše technickou záležitostí a bezpečnost je závislá hlavně na kvalitě použité šifry. V kryptografii však nikdy nejsou věci takové, jaké se na první pohled zdají a i o tom se v této sadě přesvědčíte. Ukážeme si tedy několik druhů útoků.

Přirozeně prvním je **Ciphertext-only** útok. V tomto případě má útočník k dispozici pouze zašifrovanou zprávu a jeho úlohou je zprávu dešifrovat.

Možná překvapivé je zjištění, že s tímto útokem se v praxi potkáme málokdy. Daleko častěji se totiž stává, že útočník má o naší šifře dodatečné informace. Dobrým příkladem je snaha britských kryptoanalytiků o prolomení německé Enigmy. Denně zachytávali množství zpráv šifrovaných předem domluveným klíčem. Německé jednotky posílaly denně hlášení, ve kterých byla část obsahu známá a neměnná ("Na západní frontě nic nového.") Britští analytici tedy měli k dispozici několik šifrovaných zpráv, jejichž obsah znali, a pomocí nich se snažili prolomit další. Tento typ útoku nazýváme **Known-plaintext attack**.

Třetí typ útoku je **Chosen-plaintext attack**, při kterém má útočník možnost zašifrovat libovolnou zprávu. Typickým příkladem by byl špión, který by se během obědové pauzy dostal k správně nastavené Enigmě a měl možnost na ní zašifrovat několik libovolných zpráv (denní klíče si němečtí důstojnici samozřejmě brali na oběd s sebou).

A čtvrtý druh útoku je **Chosen-ciphertext attack**, při kterém má náš špión možnost zvolit si libovolný šifrovaný text a dešifrovat ho. S pomocí získaných informací potom mají kryptoanalytici podstatně jednodušší úlohu při dešifrování dalších zpráv.

Zadání 4. sady úloh KSI (termín odevzdání: 24. 3. 2013)

Řešení zasílejte pomocí internetového systému na adrese `http://ksi.fi.muni.cz`.

Příklad 1: Římská (10 bodů)

Píše sa rok 48 p.n.l. a v Rímskej ríši vrcholí súboj o moc medzi členmi triumvirátu. Mladý cisár stojí u Farsálu proti početnej presile vojsk jeho soka Gnaeusa Pompeiusa. Jeho jedinou nádejou je poslať správu o pomoc do blízkej pevnosti. Avšak, v okolí sa to hemží nepriateľskými vojakmi a ak by sa správa dostala do nesprávnych rúk, nadchádzajúca bitka by bola stratená. Použije teda šikovný trik a všetky písmená v správe o pomoc posunie, tak ako keby sa v abecede nachádzali o 3 znaky ďalej. Správa je zašifrovaná a posol ju môže odniesť do pevnosti bez toho aby sa cisár bál, že ju nepriateľ prečíta. Nasledujúceho dňa zomrelo veľa vojakov, ale cisár vyhral a získal šifru, ktorú mohol do konca života používať. Nikto za jeho života neprišiel na to čo jeho zašifrované správy znamenali. Touto šifrou je zašifrovaný aj 1. príklad. Nepočítajte ale s tým, že to bude ľahké. V tomto prípade písmená nie sú posunuté o 3 znaky. A ak prídete na to ako text príkladu dešifrovať(čo sa vám podľa mňa nepodarí), potom je jeho ďalšia časť zašifrovaná ešte dvoma inými šiframi (pre istotu :P). Vašou úlohou je postupne dešifrovať text dole. Do riešenia vložte svoj dešifrovaný text a postup ako ste prišli na riešenie.

Šifrovaný text:

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

```
W P A F D E T D O Z L W M P C E T S Z D T Q C J Y P K L M F O
Y T O P D T Q C Z G L E L U K G J D Z V E P I E F N P L D L C
Z G Z F D T Q C Z F

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

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

Příklad 2: Praktická (10 bodů)

Tentokrát sme sa rozhodli vám poriadne potrápiť hlavy. Ako všetci správni paranoidní informatici aj orgovia KSI si šifrujú správy, aby naše tajné informácie nevyšli na svetlo sveta skôr, ako by to bolo potrebné. Preto sme vynašli novú šifru, a po vás chceme, aby ste ju otestovali.

Šifra by vám mala prísť poštou, a ak sa tak nestalo najdete ju v prílohach na stránkach KSI, odkiaľ si ju môžete stiahnuť a vytlačiť. Keďže na princíp šifry musíte prísť úplne sami, budete mať možnosť požiadania si o nápovedy na mailovej adrese: napoveda.ksi@gmail.com

Ale pozor za každú vzatú nápovedu bude bodová penalizácia, ktorá sa vam strhne z celkového hodnotenia úlohy. Konkrétne bude možné využiť postupne 5 napovied za každú sa strhávajú 2 body okrem poslednej, ktorá je za -1 bod. V tom e-maily pre korektnosť poprosíme uviesť meno súťažiaceho, inak vám konkrétna nápoveda nebude poskytnutá.

Ako správne riešenie odovzdávajte dešifrované heslo, klasickým spôsobom jak ostatné úlohy vo forme súboru na stránky KSI, ktorý nazvite tak jak vám vyjde heslo. Súbor nechajte prázdny, princíp šifry pisať nemusíte, ten sa nehodnotí, stačí nám len riešenie.

Příklad 3: Merkle (10 bodů)

V tretej úlohe na Vás čaká dobrý kus kryptoanalázy. Alica a Bob sú dobrí kamaráti a ako dôverníci by si radi vymienali tajomstvá bezpečným kanálom. Našťastie, Bob je amatérskym kryptografom a tak sa dohodli, že navrhne bezpečnú šifru.

Prvým problémom, ktorý musí vyriešiť, je bezpečné zdieľanie hesla. Klasickým spôsobom je používanie zdieľaného hesla, kedy by aj Alica, aj Bob, poznali jedno tajné spoločné heslo. Bob

však vie, že tento spôsob má svoje nevýhody, a tak sa rozhodol, že využije výdobytok modernej kryptografie - asymetrické šifrovanie. Nie je však úplným odborníkom a algebre nerozumie, preto si ako základ svojho protokolu na bezpečnú výmenu hesla vybral Merkleho hádanky[9]. Ide o veľmi jednoduchý a bezpečný protokol a preto si ho v krátkosti popíšeme.

Predpokladajme, že Alica a Bob potrebujú bezpečne ustanoviť tajné heslo. V prvom kroku Alica vygeneruje mnoho takzvaných hádaniek (v angličtine puzzles), ktoré majú nasledovný tvar:

```
PUZZLE#SERIAL#RANDOMKEY
```

Ide teda o jeden dlhý reťazec znakov, ktorý začína kľúčovým slovom PUZZLE nasledovaným identifikačným číslom hádanky. Toto číslo je náhodné, dlhé 6 znakov a žiadne dve hádanky ho nemajú rovnaké. Poslednou časťou reťazca je ľubovoľne dlhé náhodné heslo, ktoré bude použité na bezpečnú komunikáciu medzi Alicou a Bobom. Ako uvidíte ochvíľu, aby bol tento protokol bezpečný, každá hádanka by ideálne mala obsahovať iné heslo. Všetky tri časti sú oddelené oddeľovačom, v tomto prípade #. Konkrétnejšie, hádanka s poradovým číslom 654321 a tajným heslom "VELMIBEZPECNETAJNEHESLO"by vyzerala takto:

```
PUZZLE#654321#VELMIBEZPECNETAJNEHESLO
```

Alica takto vygeneruje n hádaniek, každú z nich zašifruje a všetky pošle Bobovi. Všimnite si, že v tomto bode nevadí, ak sa okrem Boba k hádankám dostane aj niekto iný - všetky zašifrované hádanky spolu tvoria Alicin verejný kľúč, ktorý môže pokojne zverejniť na svojej webstránke. Je však veľmi dôležité, aby Alica zverejnila hádanky iba v zašifrovanej podobe, a aby si nezašifrované uchovala - tie tvoria jej súkromný kľúč.

Bob má teda k dispozícii n zašifrovaných hádaniek, ktoré však nevie rozšifrovať. Náhodne si teda vyberie jednu z nich, a začne skúšať každý možný šifrovací kľúč. To mu iste zaberie istý čas, avšak skôr či neskôr určite vyskúša ten správny a vyrieši Alicinu zašifrovanú hádanku. To, že našiel správne riešenie vie jednoznačne určiť, lebo na začiatku je vždy reťazec PUZZLE. Teraz, keď má k dispozícii riešenie, pošle Alici naspäť poradové číslo hádanky a zvyšné správy už šifruje tajným heslom, ktoré našiel v riešení hádanky. Alica má hádanky uložené, takže keď od Boba dostane poradové číslo, nájde hádanku, ktorá mu zodpovedá a ihneď vie aj heslo, ktoré Bob použil.

Pozrime sa, na čom je vlastne založená bezpečnosť tohto protokolu. Eva, spoločný nepriateľ Alice aj Boba, má k dispozícii každú jednu správu, ktorú si títo dvaja priatelia vymenia. Eva najskôr zachytí všetkých n hádaniek, ktoré pošle Alica Bobovi. Všetky sú zašifrované, Eva sa teda zatiaľ nič nedozvedela. Následne zachytí správu od Boba, v ktorej Alici oznamuje poradové číslo hádanky, ktorú vyriešil. To však Eve stále nepomôže - má síce n hádaniek a vie, že v jednej z nich sa nachádza poradové číslo a tajný kľúč, ktorý hľadá, nevie však v ktorej. Jediné, čo môže skúsiť, je teda riešenie jednej hádanky za druhou. To jej však zaberie omnoho dlhší čas, než Bobovi riešenie jednej hádanky, a kým sa tak stane, Alica a Bob komunikujú bezpečne.

Vráťme sa teda k nášmu príkladu[10]. V prvom stĺpci tabuľky je nezašifrovaná podoba hádanky tak, ako ju má uloženú Alica. V druhom stĺpci je náhodný kľúč, ktorým je hádanka zašifrovaná a v treťom stĺpci je hádanka, ako ju dostane Bob.

Predpokladajme, že Bob si z n hádaniek vybral hádanku `86c9dfe19108d121a8deb8aa`. Začne skúšať všetky možné heslá, až raz natrafí na to správne: `9c6f577`. Vtedy získa text hádanky: `PUZZLE#135246#TAJN3H3SL0`. Alici následne pošle poradové číslo hádanky, teda `135246`, a ďalej už správy šifruje pomocou tajného hesla `TAJN3H3SL0`, ktoré našiel v hádanke. Alica podľa poradového čísla nájde heslo, ktoré Bob používa a obaja komunikujú bezpečne.

Všimnime si teda niekoľko faktov:

[9] V angličtine Merkle puzzles

[10] Tento príklad slúži iba na ilustráciu fungovania Merkleho hádaniek.

Alicine hádanky	kľúč, ktorým je hádanka zašifrovaná	výsledná podoba
...	...	...
PUZZLE#135246#TAJN3H3SL0	9c6f577	86c9dfe19108d121a8deb8aa
PUZZLE#086654#S3CRETK3Y5	a5cc75c	1097d538837c48b9f57dcb68
...	...	...

1. Šifra, ktorú Alica používa na šifrovanie hádaniek, musí byť bezpečná. Skúšanie všetkých možných tajných kľúčov by malo byť jedinou možnosťou, ako hádanku rozlúštiť.

2. Alica pri šifrovaní hádaniek nesmie použiť príliš bezpečné heslo, inak by ich Bob nerozlúštil. Pred začiatkom sa teda obaja dohodnú, že budú použité heslá dĺžky k znakov, a teda maximálny počet pokusov, ktoré Bob spraví, je m.

3. Bezpečnosť šifry a jej asymetria spočíva v tom, že Bobovi stačí m pokusov, avšak Eva bude potrebovať maximálne $n.m$ pokusov. Pri vhodne zvolených n a m teda bude heslo istý čas bezpečné.

4. Kedže dĺžka poradového čísla hádanky bola stanovená na 6 znakov, maximálny počet hádaniek je $n = 10^6$.

Takto teda vyzerá protokol, ktorý sa Bob rozhodol použiť. Stále však chýba niekoľko dôležitých parametrov. Aby bolo možné v rozumnom čase vyriešiť hádanku, rozhodol sa, že dĺžka použitých kľúčov na šifrovanie hádaniek bude $k = 7$. Za šifru, ktorou budú hádanky zabezpečené, si vybral Autoclave[11]. Táto šifra funguje nasledovne: text, ktorý chceme zašifrovať, zreťazíme za tajné heslo a následne použijeme na šifrovanie pôvodného text písmenko po písmenku. Za všetko hovorí obrázok:

```
TAJNA SPRAVA KTORA NEBUDE NIKDY ROZLUSTENA
HESLO TAJNAS PRAVA KTORAN EBUDE NIKDYROZLU
AEBYO SYEAOS BTJRT GSSUWR OCNHR ZYCJNJHDYU
```

Samotné šifrovanie písmenok je rovnaké ako u Vigenèrovej šifry.

Spokojný s výsledkom svojho úsilia sa Bob rozhodol, že pre istotu zájde za predsedom obecného spolku kryptológov Karlom, aby sa uistil, že jeho výsledok je ozaj bezpečný. Na jeho obrovské sklamanie mu však Karol oznámil, že Eva dokáže nájsť hádanku pomocou jej poradového čísla bez znalosti kľúča, ktorou bola hádanka zašifrovaná !

a) Ukážte, ako funguje útok, ktorý popísal Karol. K dispozícii máte n zašifrovaných hádaniek a poradové číslo hádanky. Vašou úlohou je zistiť, ktorá hádanka má dané poradové číslo bez toho, aby ste zistili kľúč, ktorým bola zašifrovaná.

Bob bol sklamaný týmto výsledkom - narobil si iba peknú hanbu ! Vzdať sa však nemienil a pokračoval v návrhu. Slabým miestom jeho protokolu sa ukázala voľba šifry. Miesto Autoclave sa teda rozhodol použiť niečo bezpečnejšie. Tentokrát už nechcel nechať nič na náhodu, preto sa rozhodol použiť OneTimePad, teda najbezpečnejšiu, ba dokonca jedinú dokázateľne bezpečnú šifru. Samozrejme je stále limitovaný dĺžkou hesla 7, no potrebuje zašifrovať

[11] V angličtine tiež Autokey cipher. Šifra objavená kryptografom Blaise de Vigenère. Naopak, Vigenèrova šifra bola objavená kryptografom Giovan Battista Bellaso.

hádanku, ktorá je omnoho dlhšia. Rozhodol sa teda využiť CBC mód, známy z blokových šifier, ktorý je takisto dokázateľne bezpečný.

Spokojný s výsledkom a istý úspechom opäť zašiel za Karlom. Ten mu na jeho zdesenie však dal rovnakú odpoveď !

a) Ukážte, ako funguje útok pri použití OneTimePad s dĺžkou hesla 7 v CBC móde. Opäť máte k dispozícii n zašifrovaných hádaniek a poradové číslo hádanky. Vašou úlohou je zistiť, ktorá hádanka má dané poradové číslo bez toho, aby ste zistili kľúč, ktorým bola zašifrovaná.

Nešťastný Bob bol dlho zúfalý, kým sa rozhodol, že skúsi šťastie ešte tretíkrát. Nič bezpečnejšie než OneTimePad nepozná, preto sa rozhodol, že chyba je vo zvolenom CBC móde. Rozhodol sa ho teda nahradiť randomizovaným CTR módom. S obavami zašiel opäť za Karlom, ktorý mu dal opäť rovnakú odpoveď.

a) Ukážte, ako funguje útok pri použití OneTimePad s dĺžkou hesla 7 v randomizovanom CTR móde. Opäť máte k dispozícii n zašifrovaných hádaniek a poradové číslo hádanky. Vašou úlohou je zistiť, ktorá hádanka má dané poradové číslo bez toho, aby ste zistili kľúč, ktorým bola zašifrovaná.

Po tomto pokuse sa Bob zaprisahal, že už nikdy nebude navrhovať vlastné šifry. Nič z toho, čo o kryptografii vedel, mu nepomohlo, naopak, ak by nešiel za skúseným kryptológom, používal by šifrovanie s falošným pocitom bezpečnosti. Uznal, že kryptografia je veľmi nevyspytateľná a odvtedy každému radí (aj Vám!), aby sa nikdy nepokúšal vymyslieť vlastné šifrovanie. Nikdy.

BONUS: napriek rade Boba sa pokúste zamyslieť, aké šifrovanie by ste zvolili Vy. Aké vlastnosti by mala Vami zvolená šifra mať, aby boli hádanky pred Evou bezpečné ? Skúste vymyslieť vlastné riešenie a nespoliehajte sa pritom na žiadne známe šifry, ako je DES či AES. Ako myslíte, že je bezpečné Vaše riešenie ? Ak k riešeniu priložíte aj túto časť, môžete za príklad dostať ľubovoľný počet bodov navyše.

Příklad 4: Padding (10 bodů)

Bob je tiež správca serveru, na ktorom by rád zverejňoval tajné dokumenty, ktoré by boli prístupné len jeho kamarátke Alici. Po skúsenostiach z prechádzajúcej úlohy sa však nepokúšal vymyslieť nič nové a zašiel za Karlom. Ten mu ako bezpečnú metódu šifrovania odporučil AES v CBC móde. Bob bol spokojný - vie, že ide o bezpečnú šifru a navyše je implementovaná v mnohých kryptografických knižniciach. Bohužiaľ, netrvalo dlho a Eva našla spôsob, ako dešifrovať ľubovoľnú správu posielanú z Bobovho servru! Ako uvidíme, aj dobre mienené vylepšenie môže mať katastrofálne následky.

Pri používaní blokových šifier sa častokrát stáva, že správa, ktorú máme zašifrovať, nie je násobkom dĺžky bloku. Aby bolo možné zašifrovať posledný blok, je nutné ho doplniť bajtami na správnu dĺžku. Pri šifrovaní typicky pridáme pár bajtov vopred dohodnutou metódou tak, aby po dešifrovaní príjemca vedel presne určiť, koľko bajtov je nutné odobrať. Aj vymýšľanie vlastného paddingu je nebezpečné, takže aj ten býva súčasťou kryptografických štandardov. Metóda, ktorú si vybral Bob, je popísaná v štandarde PKCS 5 a funguje nasledovne:

- Ak správe chýba 1 bajt, doplň bajt `01` [12]
- Ak správe chýbajú 2 bajty, doplň `02 02`
- Ak správe chýbajú 3 bajty, doplň `03 03 03`

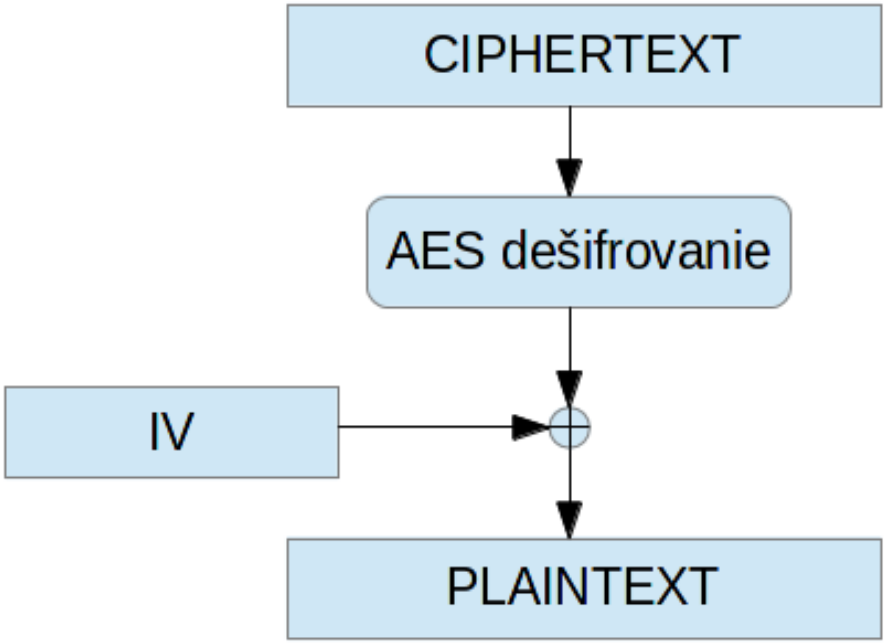
[12] použitý hexadecimálny zápis

- Ak správe chýbajú 4 bajty, doplň `04 04 04 04`
- ...

V prípade, že správa už je násobkom dĺžky bloku, musíme doplniť jeden blok s paddingom navyše. Konkrétne, ak by bola dĺžka bloku 8 bajtov, pridali by sme na koniec správy 8 bajtov(teda celý blok) s hodnotou 08.

Bob je poctivý programátor, preto vie, že na jeho server môže prísť správa, ktorá by po dešifrovaní neobsahovala správny padding. To sa môže stať napríklad ak klient, ktorý sa pripojil na server vôbec nepoužíva padding (prípadné inú verziu), je preto vhodné upozorniť ho na to chybovou správou. Bobova implementácia teda ohlási chybu klientovi v prípade, že po dešifrovaní zistí neplatný padding, tzn. ak koniec správy nie je tvorený N bajtami s hodnotou N. Príkladom neplatného paddingu je `05 05 05 FA 05`.

Práve chybovú hlášku v prípade chybného paddingu môžme využiť na úplné dešifrovanie ľubovoľného bloku. Predpokladajme, že sme zachytili správu poslanú medzi Alicou a Bobovým serverom. Ukážeme, ako je možné dešifrovať ľubovoľný blok z tejto správy, zvyšok správy dešifrujeme rovnakým spôsobom. Vezmime teda náš blok, pridajme pred neho jeden blok obsahujúci náhodné bajty a pošlime takúto správu serveru. Keďže používame AES v CBC móde, prvý blok je inicializačný vektor a druhý blok tvorí správu. Server teda dešifruje správu nasledovne:



Vidíme, že po dešifrovaní je blok XORovaný s inicializačným vektorom, ktorý sme zvolili my, a následne server skontroluje, či výsledok obsahuje platný padding. Samozrejme, s najvyššou pravdepodobnosťou sme zvolili IV tak, že padding nebude platný, a server vráti chybu. Môžme však skúsiť nájsť taký IV, s ktorým by správa mala po dešifrovaní platný padding, a server by nevrátil chybovú hlášku! Algoritmus je nasledovný:

Algoritmus 1 Dešifrovanie posledného bajtu 1

Vstup: n - dĺžka bloku v bajtoch
  vygeneruj $n-1$ náhodných bajtov $IV_1 \dots IV_{n-1}$
  for all IV_n **do**
    pošli správu s inicializačným vektorom $IV_1 \dots IV_n$ serveru
    if server nevrátil chybu Zlý padding **then**
      našli sme správny bajt IV_n
    end if
  end for

Algoritmus teda funguje nasledovne: najskôr vygenerujeme náhodný inicializačný vektor. Následne skúšame rôzne hodnoty posledného bajtu a skončíme vtedy, ak server nevráti chybovú hlášku. Keďže algoritmus skúša všetky možné hodnoty posledného bajtu, určite raz natrafí na taký, ktorého XOR s dešifrovaným blokom bude mať hodnotu `01`, a teda pôjde o platný padding. Môže sa však stať, že sme mali veľmi šťastnú ruku, a náhodné bajty inicializačného vektoru sme zvolili tak, že výsledná správa bude mať padding `02 02`, prípadne `03 03 03` či iný. Zistiť, o ktorý prípad ide je takisto jednoduché:

Algoritmus 2 Dešifrovanie posledného bajtu 2

Vstup: $IV_1 \dots IV_n$
  for $i = 1 \dots n$ **do**
    zmeň IV_i, pošli správu s inicializačným vektorom $IV_1 \dots IV_n$ serveru
    if server vrátil chybu Zlý padding **then**
      našli sme dĺžku paddingu
    end if
  end for

Princípom algoritmu je postupná zmena bytov zľava. Výsledná dešifrovaná správa pozostáva z dvoch častí: dešifrovanej správy a správneho paddingu. Ak teda zmeníme bajt dešifrovanej správy, padding ostane korektný. Ak však zmeníme bajt paddingu, určite ho tým pokazíme a server vráti chybu. Ak týmto postupom dostaneme chybu pri zmene posledného bajtu, vieme, že padding má tvar `01`, ak chyba nastane pri zmene predposledného, padding má tvar `02 02`...

Ostáva nám ukázať, ako dešifrujeme bajty pôvodnej správy. To, čo sme doteraz zmenami inicializačného vektoru získali, vyzerá nasledovne:

zašifrovaná správa	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
dešifrovaná správa	?	?	?	?	?	?	?	?
inicializačný vektor	IV_1	IV_2	IV_3	IV_4	IV_5	IV_6	IV_7	IV_8
výsledok	?	?	?	?	?	?	`02`	`02`

Vidíme teda, že v tomto prípade zistenie posledných dvoch bajtov správy je jednoduché. Pre posledný bajt dešifrovanej správy vieme, že jeho XOR s bajtom IV_8 je bajt `02`:

$$M_n \oplus IV_8 = \texttt{02}$$
$$M_n = IV_8 \oplus \texttt{02}$$

Takýmto spôsobom teda vieme dešifrovať prinajhoršom posledný bajt správy, s trochou šťastia aj pár bajtov navyše, podľa toho aký padding trafíme.

Na záver príklad s konkrétnymi hodnotami pre ilustráciu. Predpokladajme, že máme zašifrovaný blok `2B 66 FD 26 1E E5 C6 CF`. Zvolíme si teda inicializačný vektor `00 00 00 00 00 00 00 00` a budeme sa snažiť zistiť, aký IV spôsobí vo výsledku korektný padding:

Serveru pošleme	Server odpovie
`00 00 00 00 00 00 00 00 2B 66 FD 26 1E E5 C6 CF`	nesprávny padding
`00 00 00 00 00 00 00 01 2B 66 FD 26 1E E5 C6 CF`	nesprávny padding
`00 00 00 00 00 00 00 02 2B 66 FD 26 1E E5 C6 CF`	nesprávny padding
`00 00 00 00 00 00 00 03 2B 66 FD 26 1E E5 C6 CF`	OK

V tomto príklade sme teda už po štyroch pokusoch našli správny inicializačný vektor. Ďalej chceme zistiť konkrétnu hodnotu paddingu:

Serveru pošleme	Server odpovie
FF 00 00 00 00 00 00 03 2B 66 FD 26 1E E5 C6 CF	OK
00 FF 00 00 00 00 00 03 2B 66 FD 26 1E E5 C6 CF	OK
00 00 FF 00 00 00 00 03 2B 66 FD 26 1E E5 C6 CF	OK
00 00 00 FF 00 00 00 03 2B 66 FD 26 1E E5 C6 CF	OK
00 00 00 00 FF 00 00 03 2B 66 FD 26 1E E5 C6 CF	nesprávny padding

zašifrovaná správa	2B 66 FD 26 1E E5 C6 CF
dešifrovaná správa	? ? ? ? ? ? ? ?
inicializačný vektor	00 00 00 00 00 00 00 03
výsledok	? ? ? ? 04 04 04 04

Mali sme teda veľké šťastie a vidíme, že zmena posledných štyroch bajtov IV bude znamenať zmenu paddingu. Ten má teda dĺžku 4 bajty a hodnotu 04 04 04 04. Vieme teda:

Vidíme, že posledný bajt dešifrovanej správy xorovaný s bajtom 03 dá výsledok 04. Hodnota posledného bajtu je teda $M_8 =$03 $\oplus$ 04, čiže 07. Pre tri predposledné bajty platí $M_{5,6,7} =$ 00 $\oplus$ 04, teda ich hodnota je 04. Týmto spôsobom sme dešifrovali štyri bajty správy, ktorá vyzerá nasledovne: ? ? ? ? 04 04 04 07.

Vašou úlohou je ukázať, ako dokončiť tento útok a dešifrovať zvyšné bajty. Konkrétne, nájdite algoritmus, ako s pomocou znalosti posledných k bajtov dešifrovanej správy získať posledných $k + 1$ bajtov, tj. ako dešifrovať o jeden viac.

Poznámka na záver: ak ste pozorne čítali, určite ste si všimli, že tento útok nie je celkom kompletný. Posledný chýbajúci krok je však iba "technický detail", preto ho uvedieme mimo popisu samotného útoku. Proces dešifrovania správy vyzerá nasledovne:

ciphertext $\xrightarrow{DEC}$ dešifrovaná správa $\oplus$ predchádzajúci blok	= pôvodná správa
ciphertext $\xrightarrow{DEC}$ dešifrovaná správa $\oplus$ náš IV	= náhodný string s paddingom

Vyššie uvedeným útokom sme sa teda dostali iba k dešifrovanej správe. Aby sme však zistili pôvodnú správu, je nutné ešte raz výsledok XORovať s predchádzajúcim blokom zašifrovanej správy.

Příklad 5: Dvojnásobné šifrování (10 bodů)

V tomto príklade sa budeme zaoberať získaním kľúča. Majme nejakú bezpečnú šifru X a k nej príslušné funkcie:

$$encryptX(m, k) = c$$

Táto funkcia vezme správu m a kľúč k a vráti nám zašifrovanú správu c.

$$decryptX(c, k) = m$$

Táto funkcia vezme zašifrovanú správu c a kľúč k a vráti nám pôvodnú správu m.

Náš kľúč k má vždy dĺžku 32 bitov a teda máme k dispozícii 2^{32} možných kľúčov. Naša šifra je natoľko bezpečná, že jedinou možnosťou ako ju zlomiť je vyskúšať všetky možné kľúče (tzv. *bruteforce* útok). To však pre dnešné počítače pri takomto množstve kľúčov nie je žiaden problém.

V snahe vylepšiť našu šifru X sme sa preto rozhodli používať dvojicu kľúčov $(k1, k2)$ nasledujúcim spôsobom:

$$encrypt2X(m, k1, k2) = encryptX(encryptX(m, k1), k2) = c$$

Správa sa teda zašifruje najskôr jedným kľúčom a výsledok sa zašifruje druhým kľúčom.

Počet možných kľúčov sa nám tak podarilo zvýšiť na $2^{32} \cdot 2^{32} = 2^{64}$. Bezpečnosť našej šifry to ale veľmi nezvýšilo. Prečo? To je úloha pre Vás.

Vašou úlohou je popísať spôsob ako získať dvojicu kľúčov $(k1, k2)$ efektívnejšie ako požitím metódy *bruteforce*. K dispozícii máte jednu správu m a k nej príslušnú zašifrovanú správu c. Posielajte nám algoritmy v podobe pseudokódu ako aj slovný popis. Zložitosť svojho algoritmu odhadnite ako maximálny počat dvojíc kľúčov $(k1, k2)$, ktorý je potrebné vyskúšať na zlomenie šifry X. Pripomíname, že zložitosť *bruteforce* útoku by bola 2^{64}, a že Váš algoritmus by mal byť lepší.

A to je z čtvrté sady KSI vše. Přejeme ti hodně úspěchů při řešení, a když budeš mít jakékoliv otázky, neváhej se na nás obrátit e-mailem na adresu `ksi@fi.muni.cz` nebo v diskuzním fóru na webových stránkách.

Termín odevzdání 4. sady úloh KSI: 24. 3. 2013

`http://ksi.fi.muni.cz`