MODY ČINNOSTI BLOKOVÝCH ŠIFER

Překvapivý útok a česká obrana

V Chipu 5/02 jsme vás informovali o novém standardu AES i o tzv. modech, tj. způsobech jeho použití. Jeden z nich, modus CBC, byl nedávno vystaven závažnému útoku – obrana je však možná.

Ve zmíněném článku jsme ukázali, že není vhodné šifrovat soubory dat v modu elektronické kódové knihy, a doporučili jsme zejména modus CBC (je to zkratka pro *Cipher Block Chaining*, tj. pro řetězení šifrového textu). Zároveň jsme upozornili na existenci čerstvých útoků na CBC, slíbili se k nim vrátit a doporučit i vhodná protiopatření. Slib právě začínáme plnit – zaměříme se na jeden důvtipný a nečekaný útok s využitím postranního kanálu, vyúsťující v odhalení celého zašifrovaného textu.

ACHILLOVA PATA POSLEDNÍHO BLOKU

Každý, kdo implementuje šifrování dat nejen v modu CBC, ale i v ostatních modech (více viz [3]), musí řešit otázku, jak šifrovat poslední, eventuálně neúplný blok dat. Standardním řešením je tzv. *padding* (doplnění). Poslední blok se prostě doplní nějakými daty a zašifruje se. Na straně příjemce je pak nutné doplnění rozpoznat a z dešifrovaného textu ho odstranit. Aby to bylo možné, doplnění probíhá standardně takto:

Mějme blokovou šifru o *b* bajtech. Potom pokud poslední blok obsahuje *b-i* bajtů, doplníme ho *i* stejnými bajty, každým právě s hodnotou *i*. Po dešifrování pak z hodnoty posledního bajtu otevřeného textu snadno zjistíme, kolik bajtů máme odebrat. (Tak zní obecné pravidlo; pro jednoduchost budeme dále hovořit jen o blocích konkrétní délky $b = 8$.) Aby to fungovalo, je ovšem nutné doplňovat i takový otevřený text, který je už předem náhodou zarovnaný na celé osmice bajtů. V tom případě přidáme jeden celý blok osmi bajtů s hodnotou 8. Tento způsob doplňování je definován v PKCS#5 (viz [2]) a je nejrozšířenější.

Na letošní konferenci Eurocrypt, konané začátkem května, předložil Serge Vaudenay překvapivý útok na tento způsob doplňování. Položil si otázku, co se stane, když dojde k chybě v šifrovém textu, jejímž důsledkem je porušení posledního bloku otevřeného textu. Jak je vidět na obrázcích 1 a 2, vzhledem ke způsobu šifrování a dešifrování je možné takovou chybu vyvolat změnou předposledního bloku šifrového textu.

Dále je zřejmé, že volbou příslušného bajtu předposledního bloku šifrového textu můžeme docílit **přesně definované změny** v posledním bloku otevřeného textu. Zkuste si třeba představit, co se stane, když poslední bajt bloku y_{N-1} změníme operací xor FF. Poslední bajt bloku otevřeného textu x_N se nám změní stejným způsobem – xor FF. Nyní záleží na tom, jak se příslušný stroj, který přijímá šifrový text, zachová při obdržení špatného doplňku. Při dešifrování chceme samozřejmě odnímat jen správný doplněk, a proto se kontroluje, zda na konci posledního bloku je všech *i* odnímaných bajtů rovno hodnotě *i*. Pokud ne, dešifrovací stroj většinou vydá odesílateli chybové hlášení. A právě to je postranní kanál, který Vaudenay využil ke konstrukci svého „útočného" orákula.

Navrhovaná obrana maskuje hodnoty použitelné pro útočníka novými neznámými klíči.

ÚTOK

Orákulum ***O*** přijímá poslední dva bloky šifrového textu (označíme je r, y) a vrací hodnotu $\boldsymbol{O}(r, y) = 1$ v případě, že souhlasí doplněk, nebo 0, když je doplněk nesprávný. Jak uvidíme, lze tohoto orákula využít pro luštění libovolné zašifrované zprávy. Vybereme ze zprávy jakýkoliv blok šifrového textu a ukážeme si, jak pomocí orákula získat jeho dešifrovanou hodnotu D(y).

Nejprve určíme poslední bajt D(y). Orákulu ***O*** budeme posílat hodnoty (r, y), přičemž r volíme my, a to tak, že postupně procházíme všechny hodnoty jeho posledního bajtu. Uvědomme si, že (r, y) je náš „umělý" šifrový text, ale orákulum ho poslušně odšifruje a zkontroluje padding. Orákulum nám většinou odpoví nulou, protože jím vypočítaný doplněk (v bloku r xor D(y)) bude špatný, ale nejpozději po 256. pokusu (v průměru po 128 pokusech) se trefíme do platného doplňku. Proč? Stačí si uvědomit, že procházením všech hodnot posledního bajtu bloku r dojdeme jistě k tomu, že r xor D(y) bude končit bajtem s hodnotou 1, tj. platným doplňkem.

Může se samozřejmě stát, že náhodou skončí i jiným doplňkem. Všechny možnosti jsou tyto: *******1, ******22, *****333, ****4444, ***55555, **666666, *7777777, 88888888 (hvězdička znamená libovolnou hodnotu). Vzhledem k tomu, že obdržení prvního doplňku je ale nejpravděpodobnější (256krát pravděpodobnější než druhého, 256^2krát pravděpodobnější než třetího atd.), trefíme se s největší pravděpodobností do otevřeného textu s doplňkem 1. Označme si teď jednotlivé bajty bloku r jako $r_1, r_2, \ldots, r_8$ a předpokládejme, že to nastalo (u jiných doplňků se postupuje analogicky), tj. víme, že v otevřeném textu $x_N = r \text{ xor } D(y_N)$ je poslední bajt roven 1, čili platí rovnice $x_{N,8} \text{ xor } r_8 = 1$. Tím máme určen poslední bajt skutečného otevřeného textu $x_{N,8} = 1 \text{ xor } r_8$!

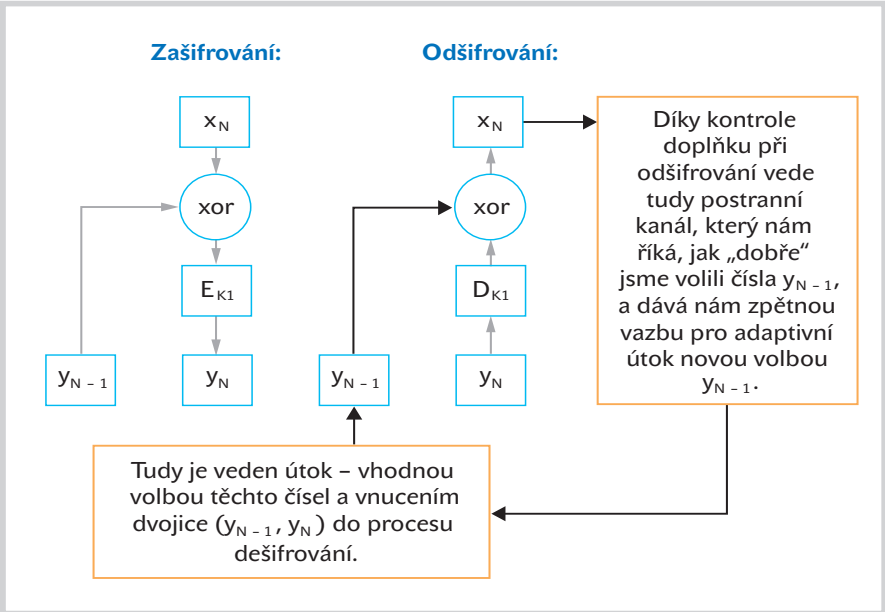
Můžeme to okamžitě ověřit tak, že definujeme $r' = r \text{ xor } (0,0,0,0,0,0,1,0)$ a odešleme (r´, y) orákulu. Oproti původní situaci orákulum zjistí otevřený text změněný v předposledním bajtu. Byl-li doplněk 1, nevadí to a orákulum vrátí opět 1.

Otevřený text: $x_1, x_2, \ldots x_N$, šifrový text: $y_1, y_2, \ldots y_N$, IV ...inicializační vektor

Zašifrování: $y_1 = E(IV \text{ xor } x_1)$, $y_i = E(y_{i-1} \text{ xor } x_i)$, $i = 2, \ldots, N$

Odšifrování: $x_1 = IV \text{ xor } D(y_1)$, $x_i = y_{i-1} \text{ xor } D(y_i)$, $i = 2, \ldots, N$

Obr. 1. Způsob šifrování a dešifrování v modu CBC



Obr. 2. Podstata útoku na modus CBC

- Byl-li doplněk jiný, změní se v něm druhý bajt od konce. Orákulum pak nutně zjistí chybu a vrátí 0.

Pomocí posledního bajtu získáme předposlední bajt takto: Pošleme orákulu (r, y), přičemž r_8 volíme tak, aby odšifrováním vznikal poslední bajt otevřeného textu 2. Toho můžeme snadno docílit volbou $r_8 = 2$ xor $x_{N,8}$, protože $x_{N,8}$ už známe. Bajt r_7 můžeme volit systematicky od 0 do 255 (a zbylé bajty r náhodně) a opět čekat na situaci, až nám orákulum odpoví jedničkou. V tu dobu musí být r_7 takové, že dva poslední bajty odpovídajícího otevřeného textu jsou 22. Čili platí rovnice $x_{N,7}$ xor $r_7 = 2$, odkud máme $x_{N,7} = 2$ xor r_7! Takto postupujeme dále, až dostaneme celý blok původního otevřeného textu x_N.

Jistě uznáte, že jde o mistrovské využití postranního kanálu u symetrických šifer. Svým významem se určitě řadí na stejnou úroveň jako postranní kanály v Mangerově a Bleichenbacherově útoku na některé implementace RSA. Všem zainteresovaným (implementátorům a uživatelům kryptografických služeb) rozhodně doporučujeme prostudovat celý příspěvek. A bude jich v tomto případě asi dost, protože modus CBC je, jak jsme už poznamenali, nejpoužívanějším modem blokových šifer.

Obrana

Vaudenay původně navrhl danou zprávu M doplnit a navíc připojit hašový kód, takže by se šifrovalo zřetězení hodnot M + padding + haš(M + padding). Wagner (viz [1]) však našel i v tomto určité bezpečnostní nedostatky a navíc takový návrh by byl nepřijatelný pro řadu kryptografických rozhraní (například *Microsoft Crypto API*), která počítají s tím, že zašifrováním posledního bloku otevřeného textu obdrží vždy nejvýše dva bloky šifrového textu; ve Vaudenayově modelu by se ale počet navracených bloků mohl zvýšit až na čtyři. Potíže by vznikly i při odšifrování závěrečných bloků, kdy by muselo být o několik bloků dříve než dnes jasné, že se jedná o koncové bloky. Proto jsme s kolegou Tomášem Rosou navrhli jiné řešení, které z vnějšího hlediska (programátorského, aplikačního aj.) vypadá stejně jako dříve, ale liší se kryptografickým vnitřkem.

Navrhli jsme tři varianty. U základní postupujeme takto: Zpráva se doplní postaru a šifruje se stejným způsobem až k poslednímu bloku. V tomto okamžiku z použitého klíče dané blokové šifry (označme ho K1) vytvoříme další tři klíče K2, K3 a K4 a pomocí nich definujeme tzv. **zodolněné šifrování posledního bloku** podle obrázku 3. Důvody jsou prosté. Zatímco dříve byl znám přesný vliv jednotlivých vstupů (x_N, y_{N-1}, y_N) na poslední blokovou operaci, nyní je vliv těchto proměnných útočníkovi zastřen neznámými klíči K2, K3 a K4. Jsou voleny odlišně od K1 proto, aby se odpovídající šifrovací a dešifrovací transformace (E_{Ki} a D_{Ki}) zásadně lišily od šifrovací a dešifrovací transformace klíče K1, které byly použity k šifrování a dešifrování předchozích bloků. Útočník, který dříve mohl ze znalosti dvojic bloků otevřeného a šifrového textu činit závěry i o zpracování posledního bloku, tuto možnost nyní nemá.

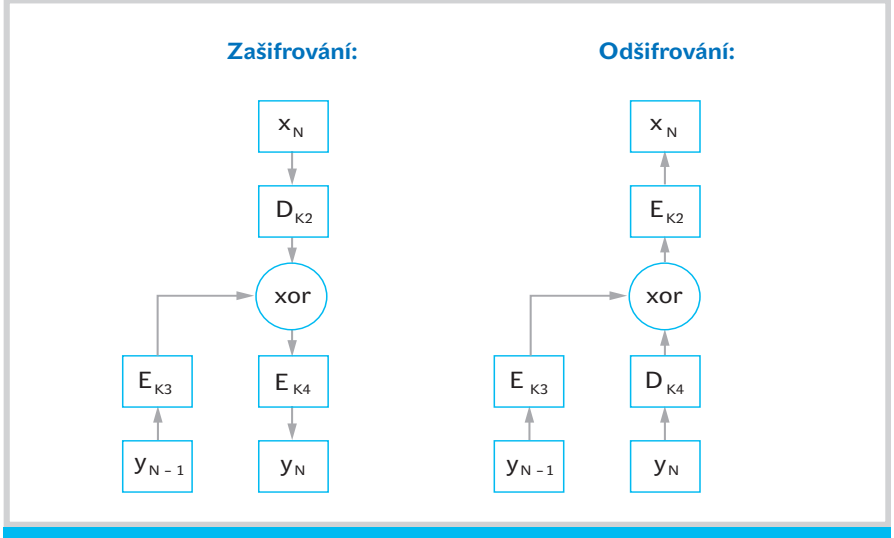
Zodolněné **dešifrování** vidíte rovněž na obrázku 3: Po odšifrování posledního bloku otevřeného textu se standardně odstraní příslušný doplněk, čili z vnějšího pohledu vše vypadá stejně jako dříve K odvození nových klíčů z K1 lze použít například normu PKCS#5 (viz např. funkce PBKDF2 v [2]), přičemž pro každý z klíčů se volí jiná hodnota tzv. soli. Je možné použít i jiné standardní postupy, důležité ale je, aby nové klíče byly odlišnými jednosměrnými deriváty původního klíče a aby i navzájem měly podobný vztah – tedy aby z jednoho klíče nešel vypočítat druhý.

Tolik ve stručnosti k námi navrženému opatření (popis dalších variant se brzy objeví v připravovaném příspěvku na internetu [4]). Poznamenejme ještě, že jak útok, tak obrana se týkají všech blokových šifer – byť by samy o sobě byly jakkoli kvalitní.

■ ■ ■ Vlastimil Klíma, *autor@chip.cz*

Literatura:

[1] Vaudenay, S.: Security Flaws Induced by CBC Padding – Applications to SSL, IPSEC, WTLS, ... , Eurocrypt 2002, pp. 534 – 545
[2] PKCS#5 v. 2.0: Password-Based Cryptography Standard, RSA Laboratories, March 25, 1999, *http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/index.html*
[3] Šifry s mnoha tvářemi, Chip 7/00, str. 50 – 53, dostupné též na [4]
[4] Archiv článků: *http://www.decros.cz/bezpecnost/_kryptografie.html*



Obr. 3. Navrhovaná obrana