OCB – velice užitečná novinka

Vlastimil Klíma

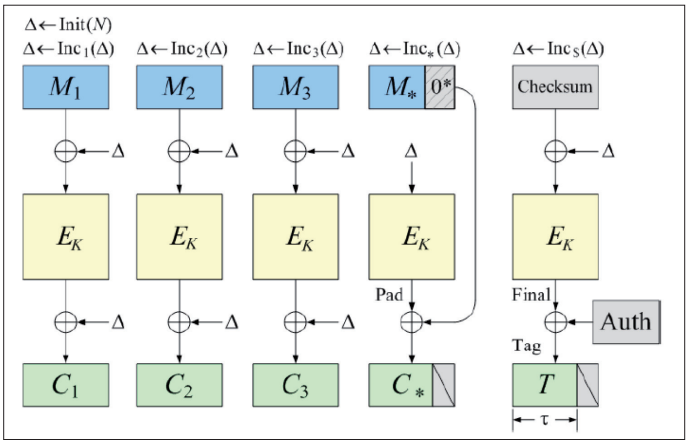
U řady aplikací, komunikačních protokolů apod. se data musí nejen šifrovat, ale připojuje se k nim i autentizační kód (MAC). To dosud vyžadovalo dva různé klíče a dvojí práci. OCB je novinka, která stihne obojí a stačí jí jen jeden klíč a jen poloviční práce. Platí se za to nevelkým nárůstem operační paměti.

Checksum jako xor?!

OCB je modus činnosti blokové šifry, při němž se zpráva současně šifruje a současně s tím se počítá autentizační kód šifrovaných dat (MAC). MAC se počítá až nevkusně jednoduše jako prostý xor bloků zprávy (Checksum), s jedinou operací šifrování „navíc“ na závěr. Nejen to, ale i celý návrh působí, jakoby OCB navrhl laik. Ale opak je pravdou, je to velice sofistikovaný návrh od těžkých teoretiků kryptologie, umně využívající léty prověřené triky neobvyklým způsobem, což umožňuje jednoduchost a bezpečnost zároveň. Ovšem pozor, OCB vznikal postupně deset let a váhu má každé slovíčko v definici! Jediný krok vedle ve stylu „tady to trochu ohnu, aby se to lépe programovalo“ – a má to katastrofální následky pro bezpečnost. Ukážeme si hlavní myšlenku, abyste zjistili, zda OCB je pro vás to pravé či ne. Podrobnosti naleznete v citované literatuře, včetně zdrojových kódů, testovacích vektorů a přesných definic. Hlavní princip současného šifrování a výpočtu MAC je vidět na *obr. 1*.

Vlastní činnost

Tak jak to probíhá? Abychom zjednodušili popis, uvažujeme blokovou šifru AES (se 128bitovým blokem) a 128bitovým klíčem (K). Zašifrování bloku je vidět na *obr. 1* jako operace E_K. Když budeme šifrovat nějaká data, je jedno, jestli jsou to data proudící na komunikačním kanálu, nebo je to flash paměť nebo pevný disk, je to prostě zpráva *M*, kterou rozdělíme na celistvé bloky po 128 bitech M_1, M_2, M_3... a možná nám zbude poslední neúplný blok M_*. Na začátku si řekněme, že počet bloků zprávy *M* nesmí být větší než 2^{64}, na *obr. 1* to jsou tři bloky a čtvrtý neúplný. Také je důležité, abychom pro každou novou zprávu (zpráva končí tehdy, pokud vydáme její autentizační kód) měli k dispozici nějaké 96bitové unikátní číslo *N* (tzv. nonce). Může to být náhodné číslo (jako je inicializační vektor u modu CBC) nebo čítač, ale musíme za každých okolností zajistit, že je unikátní pro danou zprávu. To můžeme v praxi zajistit kombinováním binární náhody, čítače ve flash paměti a třeba ještě reálným časem, a vše to poskládat do řetězce *N*. Z čísla *N* vypočítáme hodnotu $\Delta = Init(N)$, přičemž *Init* je funkce jen o málo složitější než prosté zašifrování řetězce *N*. Každý blok zprávy nyní zašifrujeme blokovou šifrou, s tím rozdílem, že ho před a po zašifrování ještě zašumíme číslem delta operací xor, jak ukazuje *obr. 1*. No, a číslo delta se před každým použitím vždy aktualizuje funkcí *Inc*. Můžeme si ji představit jako skutečné inkrementování, není o mnoho složitější. Funkce *Inc* také používá jako vstup index (jakoby to bylo inkrementování počáteční hodnoty o „index“ bloku), ale je to v podstatě xor s některou z tabulkových hodnot L(0), L(1)...L(63), přičemž se jedná o konstanty, nezávislé na klíči. Konstant musíme uchovávat tolik, kolik je logaritmus maximální délky zprávy, kterou chceme naším zařízením (systémem) kdy šifrovat. Je-li maximum 2^{32} bloků, pak potřebujeme 32 konstant (+2 zvláštní), nejvíce však 64 konstant (+2). Tyto paměťové nároky se zdají být jedinou nevýhodou OCB. Předtím, než se blok zprávy zašumí, přixoruje se do Checksum. Pro jednoduchost uvažujme, že máme jen celistvé bloky, takže jakmile máme poslední zašifrovaný, můžeme už vypočítat MAC, a to jednoduše tím, že zašumíme Checksum pomocí delta (zde se delta inkrementuje jednou ze zvláštních konstant L$), a tuto hodnotu pak zašifrujeme. Přixorování hodnoty Auth na *obr. 1* je další možnost, kdy nějaká data nejsou šifrovaná, ale jen autenizovaná, z nich se počítá určitým způsobem kód Auth, a ten se pak přičítá na konci, jak je vidět na obrázku. Data, která se nešifrují, ale jen autentizují, jsou v protokolech velice důležitá, přičemž OCB na to pamatuje. Pokud taková data nejsou, Auth je nulový. No, a z výsledku na obrázku se vezme tolik bitů (tau), kolik potřebujeme a máme autentizační kód (T) celé zprávy *M!*



Obr. 1 Princip OCB – geniální metoda současného šifrování a výpočtu autentizačního kódu

Hlavní trik

Hlavní trik OCB spočívá v zašumění. Pokud by nebylo zašumění, Checksum by nerozpoznal, když by útočník vyměnil dva bloky šifrového textu mezi sebou. Takhle to pozná, protože když se bloky dešifrují na nesprávné pozici, použije se na ně nesprávné zašumění hodnotou delta, která je jiná na každé pozici. Tím se obdrží „posunutá“ hodnota šifrového textu, která při dešifrování vede na náhodný otevřený text, a ten pochopitelně v Checksum způsobí náhodný chaos.

OCB neprodlužuje šifrový text!

Důležitou vynikající vlastností OCB je, že šifrový text je stejně dlouhý jako otevřený text! Tedy kromě hodnoty MAC, ovšem na něj se předpokládá pevné místo. Je to tedy výhodnější než oblíbený modus CBC, který otevřený text vždy doplňuje (i kdyby byl zarovnaný), a tudíž vždy natahuje šifrový text. Zde se na poslední neúplný blok prostě naxoruje tolik bajtů hesla, jaká je délka tohoto bloku, přičemž heslo se získá zašifrováním hodnoty delta, (inkrementované o druhou zvláštní konstantu L*)! A neúplný blok se opět prostě naxoruje do Checksum! To vše vidíme jednoduše na *obr. 1* u posledního bloku M_*. Ještě poznamenejme na závěr, že Auth se z nešifrovaných dat počítá velice podobným postupem, jak byl popsán výše, čili nepřibývá žádná nová složitost.

OCB šetřílek

Dosud zavedené standardy požadují šifrovat data jedním klíčem a počítat jejich autentizační kód jiným klíčem. Navíc oba procesy jsou stejně náročné, takže jakoby musíme data dvakrát šifrovat. Ještě, že to můžeme dělat souběžně a nemusíme čekat na konec procesu šifrování celých dat, to by bylo nepoužitelné. OCB však spoří čas i výkon, ale bere si za to trochu paměti na konstanty L(0...63).

LITERATURA

[1] *http://www.cs.ucdavis.edu/~rogaway/ocb/index.html.*

[2] *The OCB Authenticated-Encryption Algorithm, IETF, Internet draft, Jan. 2012, http://datatracker.ietf.org/doc/draft-krovetz-ocb.*