Téma 2 – Principy kryptografie

Elektronický podpis

V tomto čísle se podrobněji podíváme na to, jak funguje elektronický podpis. Ukážeme si, jak funguje podepisování založené na asymetrické kryptografii, které je využíváno například v nových občanských průkazech nebo při použití klientských certifikátů na webu. Ale to nám nebude stačit, a tak se budeme zabývat i implementačně jednodušší formou ověření integrity zprávy, která je založena čistě na hashovacích funkcích.

Pokud jste v rámci tohoto tématu nečetli úvodní texty k hashovacím funkcím (vyšlo v 2. čísle) a k asymetrické kryptografii (najdete v 4. čísle), doporučujeme je před čtením tohoto textu alespoň zběžně prolistovat.

Co mám podepsat?

Asi už tušíte, že pokud v nějaké kryptografické operaci drobně upravím vstup, mívá to zásadní dopad na výsledek. V mnoha případech je takové chování dokonce velmi žádoucí. V případě elektronického podpisu to na nás ale klade velké nároky na přesnost: podpis bude vždy platný pouze pro přesně specifikovaný dokument. Pokud na jeho konec přidám navíc jeden prázdný řádek, podpis se stane neplatným. Než nějaká data podepíšu, musím přesně specifikovat, jak vypadají.

Pokud chci podepsat jenom nějakou skupinu hodnot, stačí mi definovat, jak je za sebe poskládám.

Problém 1 [1b]: *Představme si, že v elektronickém bankovnictví chceme podepsat příkaz k úhradě. Nejjednodušší přístup je vzít jednotlivé položky platby a naskládat je za sebe.*

Výsledný řetězec by mohl mít například tvar $UCET\_ODESILATELE \| UCET\_ADRESATA \| ZPRAVA\_PRO\_ADRESATA \| CASTKA \| KONSTANTNI\_SYMBOL \| VARIABILNI\_SYMBOL \| SPECIFICKY\_SYMBOL$*, kde* $\|$ *značí spojení řetězců.*

Je takový přístup pro reálné situace dostatečný, nebo přináší nějaké riziko? Dokážete navrhnout, jak vytvořit řetězec lépe?

Trochu komplikovanější je situace ve chvíli, kdy chceme podepsat celý dokument. Mohli bychom vzít binární soubor tak, jak je uložený na disku počítače, a k němu spočítat podpis. To se skutečně často dělá. Ale ve výsledku pak máme dva soubory – vlastní soubor s obsahem a podpis, což nemusí být praktické. Pravděpodobně jste někdy potkali podepsaný PDF dokument a žádný speciální soubor s podpisem jste nepotřebovali. To je možné díky tomu, že formát PDF (a mnohé další) počítá s možností uložení podpisu přímo do dokumentu. V rámci dokumentu je specifikovaná oblast, která je podepisovaná, a vedle je uložený podpis.

Napadá vás nějaké úskalí tohoto přístupu? Když máte podepsaný PDF dokument, není podepsaný celý dokument, ale pouze jeho část! Mnoho prohlížečů vám ale pouze zobrazí informaci o tom, že dokument je podepsaný, ale už nespecifikuje, co všechno podepsaná část obsahuje. Takže to může být třeba jenom nadpis.

Asymetrická kryptografie

Všimli jste si v minulém čísle zvláštní symetrie u RSA? Když šifruji pomocí veřejného exponentu (e), dělám úplně ty samé operace jako když pomocí privátního exponentu (d) dešifruji. Tato skutečnost nám umožňuje využít RSA „naopak“ pro elektronický podpis.

Pokud chci nějaký dokument podepsat, tak ho zašifruji pomocí privátního exponentu (d) a dokument i šifrový text, kterému nyní říkám *podpis*, zveřejním.

Každý, kdo zná můj veřejný klíč (n,e), si nyní může můj podpis ověřit. Pokud podpis dešifruje pomocí veřejného exponentu (e), dostane zveřejněný dokument.

Pro praktické použití je omezením skutečnost, že pomocí RSA nemohu šifrovat zprávy větší, než je modul (značili jsme n). Proto se typicky nepodepisuje přímo zpráva, ale její hash. Z dílu o hashovacích funkcích víme, že jsou navrženy tak, aby bylo výpočetně velmi obtížné najít dvě zprávy se stejným hashem. Pokud tedy dostaneme zprávu a podpis hashe, který zprávě odpovídá, je velmi pravděpodobné, že byla podepsána skutečně tato zpráva.

Pokud někdy uslyšíte o podepisování pomocí certifikátů (třeba i generovaných na různých čipových kartách a podobně) a bude vám to připadat hrozně složité, věřte, že za tím bývá často právě RSA, případně ne o moc složitější algoritmy nazývané DSA či ECDSA.

Problém 2 [3b]: *Pokud chceme uzavírat elektronicky smlouvy, objevuje se nám ještě jeden problém: často je důležité prokázat, kdy byla smlouva uzavřena. Napadá vás nějaké řešení, jak do podpisu přidat nezpochybnitelnou informaci o času, kdy k němu došlo?*

Problém 3 [1+10b]: *Ze stránek tématu* `https://mam.mff.cuni.cz/problem/2203/` *si můžete stáhnout PDF podepsané pomocí RSA. Zkuste zjistit co nejvíce o použitém klíči. Jeden bod je za modul a veřejný exponent. Za privátní klíč je speciální odměna 10 bodů. Pokud ho budete chtít získat, asi se budete muset porozhlédout po nějakých známých útocích na RSA.*

Problém 4 [2b]: *Chcete poslat řešení úlohy obsahující libovolné nesmysly a získat plný počet bodů? Právě máte šanci. Za správné bude pokládáno každé řešení, které bude zasláno elektronicky v PDF podepsaném klíčem, který najdete na webových stránkách tématu.*

HMAC

Elektronický podpis je prostředkem, jak zaručit integritu posílané zprávy. Pokud se spolu baví dvě strany a chtějí mít jistotu, že zasílané zprávy nikdo po cestě nijak neupraví, stačí, když budou obě strany odesílané zprávy podepisovat a druhá strana bude podpisy ověřovat.

Ve chvíli, kdy si komunikující strany navzájem věří, je využití asymetrických podpisů zbytečně složité. Stačí využít vlastností nám již známých hashovacích

funkcí. Ty by měly fungovat tak, aby znalost hashe nedávala žádnou informaci o původním textu. Dokonce i když znám kus původního textu, tak by mi znalost hashe neměla pomoct poznat i zbylou část.

HMAC (Keyed-hash Message Authentication Code) v principu funguje tak, že si komunikující strany nejdříve vymění symetrický sdílený klíč (tedy obě strany mají stejný klíč) a potom počítají hash z klíče spojeného se zprávou.

Problém 5 [2 + 4b]:

Nejprimitivnějším přístupem pro tvorbu HMAC by bylo počítat Hash(klíč $\|$ *zpráva), kde* $\|$ *značí spojení textových řetězců.*

Takový přístup by ale při použití běžných hashovacích funkcí (SHA1, SHA256, RIKSHA, ...) umožňoval při odposlechu podepsaných zpráv podepisovat další zprávy i bez znalosti klíče. Za teoretický popis slabiny nabízíme 2 body.

Další 4 body lze získat za praktickou demonstraci. Zachytili jste zprávu „Mam rad jablka.“ a její podpis 50BA1E0A5660AC90. Víte, že podpis je spočítán jako RIKSHA(klíč $\|$ *zpráva). Dokážete získat podpis zprávy, která obsahuje mimo jiné text „Nemam rad hrusky“?*

Popis hashovací funkce RIKSHA včetně její kompletní implementace lze najít ve 2. čísle.

Aby bylo zamezeno praktickým i spíše teoretickým útokům, počítá se MHAC dle standardu trochu složitěji, konkrétně:

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left((K \oplus opad) \,\|\, \mathrm{H}\left((K \oplus ipad) \,\|\, m\right)\right),$$

kde H je hashovací funkce, K tajný klíč zarovnaný na velikost bloku, m podepisovaná zpráva a $opad = 5C5C\ldots5C$ a $ipad = 3636\ldots36$ dvě fixní konstanty dlouhé dle velikosti bloku hashovací funkce.

Ačkoli na první pohled může HMAC vypadat složitě, je jeho výpočet velmi rychlý a ve chvíli, kdy je potřeba zajistit integritu velkého množství dat, je velmi užitečný.

Problémy zadané v předchozích číslech

Až do termínu odeslání páté série můžete posílat také řešení všech problémů zadaných ve třetím a čtvrtém čísle a úloh 4, 5 a 6 z druhého čísla.

Pokud jste již nad těmito problémy strávili spoustu času a zajímalo by vás řešení, tak se nebojte, že byste se ho nedočkali. Vzorová řešení všech zadaných úloh otiskneme.

Kuba a Káťa; jakub.topfer@matfyz.cz
e-mailová konference: krypto@mam.mff.cuni.cz