# Digitální podepisování

Kateřina Pastirčáková

Úvod do kryptologie

10. dubna 2014

# Trocha historie

- Egypt - Kartuš, otisk Scarabea
- Asie - kamenná pečetítka
- Řekové - pečetní prsten
  - uzavírá
    - nepřístupné nepovolaným
    - chrání proti falšování
  - ověřuje právní platnost
- Renezance a rozšíření písma - podpis
- 19. století - telegrafický podpis
- 1980 - faxem obdržený podepsaný dokument
- konec 80. let - elektronický podpis

# Požadavky na podpis

- **Autenticita**
  - lze ověřit identitu odesilatele
- **Integrita**
  - zpráva nebyla od podepsání změněna
- **Nepopiratelnost**
  - autor nemůže tvrdit, že elektronický podpis příslušný k dokumentu nevytvořil
- **Časové ukotvení**
  - doklad o tom, kdy byl dokument podepsán

# Jak na to?

Využijeme **asymetrickou šifru** a **hashovací funkci**

1. Vygenerujeme hash zprávy
2. Hash zašifrujeme pomocí našeho **soukromého klíče**
3. Příjemce si zjistí náš veřejný klíč (standardně od důvěryhodné třetí strany, případně mu jej pošleme spolu se zprávou - autenticita?)
4. S jeho pomocí dešifruje hash a porovná ji s hashí přijaté zprávy
    - Integrita

# Jak na to?

Využijeme **asymetrickou šifru** a **hashovací funkci**

1. Vygenerujeme hash zprávy
2. Hash zašifrujeme pomocí našeho **soukromého klíče**
3. Příjemce si zjistí náš veřejný klíč (standardně od důvěryhodné třetí strany, případně mu jej pošleme spolu se zprávou - autenticita?)
4. S jeho pomocí dešifruje hash a porovná ji s hashí přijaté zprávy
   - Integrita
   - Nepopiratelnost

# Jak na to?

Využijeme **asymetrickou šifru** a **hashovací funkci**

1. Vygenerujeme hash zprávy
2. Hash zašifrujeme pomocí našeho **soukromého klíče**
3. Příjemce si zjistí náš veřejný klíč (standardně od důvěryhodné třetí strany, případně mu jej pošleme spolu se zprávou - autenticita?)
4. S jeho pomocí dešifruje hash a porovná ji s hashí přijaté zprávy
   - Integrita
   - Nepopiratelnost
   - Autenticita ?

# Jak na to?

Využijeme **asymetrickou šifru** a **hashovací funkci**

1. Vygenerujeme hash zprávy
2. Hash zašifrujeme pomocí našeho **soukromého klíče**
3. Příjemce si zjistí náš veřejný klíč (standardně od důvěryhodné třetí strany, případně mu jej pošleme spolu se zprávou - autenticita?)
4. S jeho pomocí dešifruje hash a porovná ji s hashí přijaté zprávy
   - Integrita
   - Nepopiratelnost
   - Autenticita ?
   - Časové ukotvení ?

# Public Key Infrastructure

Jak věrohodně předat veřejný klíč:

- Certifikační autorita
  - Vydává **certifikáty** = elektronicky podepsané veřejné šifrovací klíče (+další údaje o majiteli, časová platnost)
  - Potvrzuje pravdivost údajů uvedených v certifikátu
  - Odpovídá za zařazení svých kořenových certifikátů do software
  - Může vydávat také časová razítka

- Síť důvěry
  - Svůj veřejný klíč si uživatelé nechávají od důvěryhodných osob elektronicky podepisovat
  - Odesilatel předpokládá, že příjemce bude **věřit** alespoň jednomu elektronickému podpisu v jeho veřejném klíči

# Public Key Infrastructure

Akreditovaní poskytovatelé certifikačních služeb:

- Česká pošta
- První certifikační autorita
- e Identity

- Cena přes 400Kč/rok
- Zdarma – k některým bankovním účtům
- Zdarma – nekvalifikované certifikáty

# Vybudovaná infrastruktura

**BONUS – šifrované podepsané zprávy**

- Odesilatel i adresát mají svůj veřejný i soukromý klíč
- Zprávu nejprve zašifrujeme adresátovým **veřejným** klíčem
- Poté podepíšeme hash šifrované zprávy naším **soukromým** klíčem = **podpis**
- Adresát porovná hash obdržené šifrované zprávy s podpisem, který dekoduje pomocí našeho (důvěryhodného) veřejného klíče
- Svým soukromým klíčem dekoduje zprávu

# Proč používáme hash?

- Volitelná standardizovaná délka
- Rychlost
- Kompatibilita – většina šifrovacích algoritmů pracuje s čísly
- Bezpečnost
  - dvě zprávy lze spojit do jedné, na čemž mohou být založeny útoky

# Útok

- Podepisujeme jen pomocí RSA, bez hashování

## Opakování – RSA

$n = pq, \quad \varphi(n) = (p-1)(q-1),$
Veřejný klíč $(n, e), \quad 1 < e < \varphi(n), \quad e \perp \varphi(n)$
Soukromý klíč $d, \quad de + k\varphi(n) = 1$

- Podepisujeme zprávu $m$ : $\quad c \equiv m^d \mod n$
- Příjemce dešifruje $c$ : $\quad m \equiv c^e \mod n$

- **Útočník** chce poslat falzifikát $m$, vytvoří zprávy $m_1, m_2$

$$m \equiv m_1 m_2 \mod n$$

- Získá od nás podpisy zpráv $m_1, m_2$
- Tím dostane podpis falzifikátu $m$, neboť

$$m^d \equiv (m_1^d)(m_2^d) \mod n$$

# PGP/GPG

- Komerční program PGP = Pretty Good Privacy (1991)
- Jeho open source alternativa GPG = GNU Privacy Guard (1999)
- Programy pro šifrování a podepisování zpráv, emailů
- Obvykle šifrují zprávu symetrickou šifrou (rychlejší) a pouze klíč šifrují asymetricky
- Využívají
  - šifry symetrické: AES, 3DES, blowfish, CAST5, RC4
  - šifry asymetrické: RSA, ElGamal, DSA, ECDSA
  - hashovací algoritmy: MD5, RIPEMD, SHA1,2,3, Tiger

# DSA (Standard)

- Digital Signature Algorithm (1991)
- Problém diskrétního logaritmu
- Zvolíme délky klíčů $N, L$ – kde $N \geq$ délka hashe; doporučení $L \in \{2048, 3072\}$
- Generujeme $N$-bitové prvočíslo $q$
- Generujeme $L$-bitové prvočíslo $p$ tak, že $q \| p - 1$
- Generujeme $g$: multiplikativní řád $g$ v tělese $\mathbb{Z}_p$ je $q$
  - z algebry víme, že je-li prvek $a$ generátorem $\mathbb{Z}_p^*$ (těch je $\varphi(p-1)$), je $a^{\frac{p-1}{q}}$ prvek řádu $q$
  - $a^{\frac{p-1}{q}}$ může být řádu $q$ nebo 1
- Volíme náhodně privátní klíč $x, 0 < x < q$
- Veřejný klíč $(p, q, g, y)$, kde $y = g^x \mod p$

# DSA – podpis

Hash $M$ odesílané zprávy podepíšeme následovně

- Pro danou zprávu se vybere náhodná hodnota $k, 0 < k < q$
- Spočteme $r = (g^k \mod p) \mod q$
- Spočteme $s = (k^{-1}(M + xr)) \mod q$
- Je-li $r = 0$ nebo $s = 0$ (nepravděpodobné), výpočet opakujeme od začátku
- Podpisem je dvojice $(r, s)$

# DSA – ověření podpisu

Ověření podpisu $(r, s)$ přijaté zprávy s hashí $M$

- Nutně $r, s \in \widehat{q-1}$, jinak je podpis automaticky zamítnut
- Spočteme $w = s^{-1} \mod q$
- Spočteme $u_1 = Mw \mod q$
- Spočteme $u_2 = rw \mod q$
- Spočteme $v = (g^{u_1} y^{u_2} \mod p) \mod q$
- Podpis platí, pokud $v = r$

# DSA – ověření

- $s$ jsme zvolili jako $s = (k^{-1}(M + xr)) \mod q$
  - $k \equiv Ms^{-1} + xrs^{-1} \mod q$
  - $k \equiv Mw + xrw \mod q$
- prvek $g$ je řádu $q$ v $\mathbb{Z}_p^*$
  - $g^k \equiv g^{Mw+xrw} \mod p$, přičemž $y = g^x \mod p$
  - $g^k \equiv g^{Mw} y^{rw} \mod p$
  - $g^k \equiv g^{u_1} y^{u_2} \mod p$
- $r$ jsme zvolili jako $r = (g^k \mod p) \mod q$
- $r \equiv (g^{u_1} y^{u_2} \mod p) \mod q \equiv v$

# Využití

- Podepisování emailů
- Zasílání elektronických faktur
- V elektronických bankovních transakcích
- Komunikace s úřady (e-government)
- Web servery

# HeartBleed v OpenSSL

- Zabezpečení $\frac{2}{3}$ webových serverů, emaily, internetová bankovnictví... (https)
- V kodu chybí kontrola velikosti ukazatele
- Předává ukazatel na až 64kB paměti, ale nehlídá obsah paměti
- Únik hesel, emailů a šifrovaných zpráv, možná i privátních klíčů
- Opraveno 7. dubna 2014, chyba v programu byla přes rok
- Jak moc bezpečný je zdarma dostupný software...???

# Shrnutí

- Co je to elektronický podpis
- Jak jej získat
- Jak funguje
- Algoritmy

# Shrnutí

- Co je to elektronický podpis
- Jak jej získat
- Jak funguje
- Algoritmy

**Děkuji za pozornost**

# Zdroje


- en.wikipedia.org/wiki/Electronic_signature
- www.ica.cz/Elektronicky-podpis
- www.root.cz/clanky/uvod-do-gpg/
- www.princeton.edu/~achaney/tmve/wiki100k/docs/Chosen-ciphertext_attack.html
- crypto-world.info/pinkava/konference/cack.pdf
- www.scribd.com/doc/135716080/DSS-Digital-Signature-Standard-and-DSA-Algorithm
- blog.torproject.org/blog/openssl-bug-cve-2014-0160