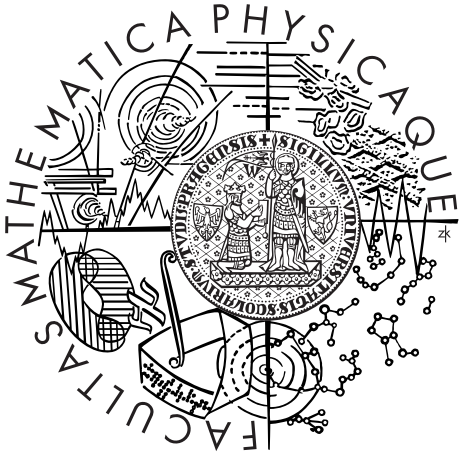
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Veronika Čadová

O DSA

Katedra algebry

Vedoucí bakalářské práce: prof.RNDr.Drápal Aleš,Csc.,DSc.

Studijní program: Matematika

Studijní obor: Matematické metody informační bezpečnosti

Praha 2011

Děkuji vedoucímu své bakalářské práce, prof.RNDr. Alešovi Drápalovi,Csc.,DSc. a Mgr. Janu Zvánovci za cenné rady, trpělivost a čas, který mi věnoval. Dále děkuji své rodině a přátelům za podporu v průběhu psaní bakalářské práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: O DSA

Autor: Veronika Čadová

Katedra: Katedra Algebry

Vedoucí bakalářské práce: prof.RNDr.Drápal Aleš,Csc.,DSc.

Abstrakt: Bakalářská práce se věnuje problematice porovnávání bezpečnosti a složitosti digitálních podpisů DSA a Schnorr. Digitální podpis je téměř plnohodnotnou, zákonem uznávanou alternativou k fyzickému podpisu, určenou pro využití v digitálním prostředí. Princip využívá asymetrických šifer a hašovacích funkcí, které jsou zde jednoduše popsány, stejně jako další základní pojmy, mezi něž patří problém diskrétního logaritmu a cyklické grupy. Práce se zabývá analýzou některých možných útoků na DSA a porovnáním DSA a Schnorrova algoritmu. Součástí textu je i pohled do historie a vlastní implementaci digitálního podpisu.

Klíčová slova: Haš, cyklická grupa, diskrétní logaritmus, DSA, Schnorr

Title: On DSA

Author: Veronika Čadová

Department: Department of Algebra

Supervisor: prof.RNDr.Drápal Aleš,Csc.,DSc.

Abstract: This thesis deals with problems of comparing the safety and running time of digital signatures DSA and Schnorr. Digital signature is almost full, legally recognized alternative to physical sign, intended for use in a digital environment. Digital signature uses asymmetric codes and hash functions which are easily described, as well as other basic concepts such as discrete logarithm and cyclic groups. The thesis deals with the analysis of possible attacks on DSA and compares DSA and Schnorr algorithm. Digital signature history and its implementation is part of the thesis.

Keywords: Hash, Cyclic Group, Discrete Logarithm, DSA, Schnorr

Obsah

Úvod

V minulosti i v současnosti jsme se s obyčejným podpisem setkávali na každém kroku. Jeho hlavní rolí je identifikace identity člověka, který se podepisuje. V běžném prostředí je přiměřeně unikátní podpis akceptován jako dostatečný důkaz identity a může být chápán jako autentizující prostředek nebo, při vyjádření stanoviska, také jako autorizující prostředek.

Již delší dobu jsou k dispozici takové technologie, které dokáží plně nahradit to, k čemu slouží tradiční podpis vlastnoruční.

Pojem „Digital signature“ je rozšířen spíše v zemích na americkém kontinentu zřejmě v návaznosti na americký Digital signatures standard. V zemích Evropské Unie je směrodatná Direktiva Evropské komise o elektronickém podpisu (Electronic Signature Directive), z níž vychází i Zákon o elektronickém podpisu platný v České republice.

Digitální podpis se z kryptografického hlediska chápe jako soustava kryptografických funkcí. Jeho bezpečnostní cíle jsou:

- *autentizace* – ověřuje, že daná entita je skutečně tou, za kterou se vydává.
- *integrita* - zajistí, aby podpis nebyl nijak pozměněn na cestě k příjemci.
- *nezfalšovatelnost* - zajistí, aby útočník nemohl vytvořit platný podpis ke zprávě bez znalosti soukromého klíče.
- *nepopiratelnost* - zajistí, aby daná podepisující entita nemohla později popřít to, co podepsala.

Digitální podpis v současné běžné podobě má základ v *asymetrické kryptografii*. Na rozdíl od symetrické kryptografie, která pro zašifrování a dešifrování dat používá stejný šifrovací klíč, asymetrická kryptografie se vyznačuje tím, že používá pro šifrování odlišný klíč než pro dešifrování. Je to *veřejný klíč* pro zašifrování dat, který může mít kdokoliv, a jeho pomocí zašifrovat data, a *soukromý klíč* pro dešifrování dat, který má v tajnosti pouze majitel, aby si mohl data dešifrovat. V případě digitálního podpisu se používá soukromý klíč k vytvoření podpisu a veřejný klíč pro jeho ověření. Důležitou podmínkou pro tyto klíče je, aby nebylo možné se znalostí veřejného klíče si spočítat soukromý klíč.

Skutečně spolehlivé doručení je však netriviální problém především v situaci, kdy komunikace s příjemci zpráv probíhá po nezabezpečené síti a bez předchozí domluvy po bezpečném kanále. Za takové situace nelze považovat jakékoliv doručení veřejného klíče příjemci za úplně spolehlivé.

Základní pojmy k porozumění digitálního podpisu jsou uvedeny v první kapitole. V dalších dvou je popsáno, jak fungují dva známé algoritmy DSA a Schnorr. Ve čtvrté kapitole jsou vysvětleny některé útoky na DSA a porovnání se Schnorovým algoritmem. V páté kapitole je popsána analýza jedno z útoků, která v učebnicích není definována. Ke konci práce najdeme něco k historii a popis vlastní implementace.

1. Základní pojmy

V této kapitole se budeme věnovat pojmům, které jsou potřebné k pochopení funkčnosti digitálních podpisů, kterými se budeme věnovat později. Čerpáme ze skript Stanovkého [1] a knih, které se věnují základním poznatkům z kryptografie [2], [3].

1.1 Hašovací funkce

Nevýhodou asymetrického šifrování je jeho malá rychlost. Jeho použití je značně pomalejší než u symetrických šifer. Při podepisování větších datových zpráv by tak uživatel strávil mnoho času čekáním na dokončení šifrování, proto nejprve zprávu, kterou chceme podepsat, hašujeme a teprve pak kratší haš podepíšeme.

Definice. *Hašovací funkce* je funkce h, která

- zobrazuje zprávu m libovolné konečné bitové délky na výstup $h(m)$ fixní bitové délky (komprese),
- umožňuje ze zprávy m lehce vypočítat $h(m)$.

U výstupu hašovací funkce v kryptografii se předpokládají následující vlastnosti:

1. *jednosměrnost* - snadno lze z m vypočítat $h(m)$, ale obráceně je to výpočetně nemožné,
2. *bezkoliznost* - výpočetně nemožné najít dvě zprávy m, m' takové, že hodnoty jejich hašů se rovnají $h(m) = h(m')$ (kolize). Také by mělo být výpočetně nemožné najít k dané zprávě jinou zprávu se stejnou hodnotou haše.

Chtěli bychom dosáhnout, aby se hašovací funkce chovala jako náhodné orákulum, které pro libovolný vstup dává náhodný výstup vždy se stejnou pravděpodobností pro všechny možné prvky výstupní množiny hodnot. Platí však, že pro stejné vstupní hodnoty dává stejné výstupní hodnoty. Současné hašovací funkce mají chování blízké náhodnému orákulu, bohužel dokonalé náhodné orákulum zatím stále chybí.

Ve skutečnosti je obtížné najít kolize, ale to neznamená, že neexistují. Hašovací funkce má krátký výstup a možných hašů je vždy jen 2^N, kde N je délka

výstupu hašovací funkce v bitech. Pravděpodobnost, že náhodně vybraný text, jehož haš bude odpovídat nějakému konkrétnímu zadanému haši, je $\frac{1}{2^N}$. Zatím nebyla nalezena prokazatelně bezpečná hašovací funkce.

Současné hašovací funkce

MD5 - Navrhl ji Rivest v roce 1991 a nahrazuje dříve používanou MD4. Výstup MD5 je 128 bitový a je spolu s SHA-1 nejpoužívanější hašovací funkcí, a to i přes objevené slabiny.

SHA-1 - Nejpoužívanější hašovací funkce publikovaná v roce 1993 nejprve jen jako SHA, v roce 1995 vylepšená a označená jako SHA-1. Původní verze se někdy označuje jako SHA-0. Funkce je standardizována organizací NIST(National Institute of Standards and Technology) v dokumentu FIPS 180 (Federal Information Processing Standard). Funkce produkuje haše délky 160 bitů.

SHA-2 - Je to „rodina“ obsahující SHA-224, SHA-256, SHA-384 a SHA-512. Byla vytvořena v roce 2001, 2002 a 2004. Číslo za názvem algoritmu značí délku výstupu v bitech. Zatím nebyla ještě tolik testována jako SHA-1, proto se více používá SHA-1.

SHA-3 - NIST vyhlásil soutěž o vymyšlení SHA-3, do které se mohly posílat účastníci příspěvky do 8. října 2008. Bylo přijato 64 příspěvků. Nyní je už vybráno 5 finalistů a vítěz by měl být zveřejněn v roce 2012. Průběh soutěže najdeme na [4].

Další funkce - RIPEDM, HAVAL, Tiger, HDN, Grindahl.

1.2 Cyklická grupa $\mathbb{Z}_p^*$

V následujícím textu si definujeme, co je cyklická grupa. Zaměříme se pouze na multiplikativní konečné grupy $\mathbf{G} = (G, \cdot, ^{-1}, 1)$ a to z důvodu, že tento text se dále bude zabývat konečnou a multiplikativní grupou $\mathbb{Z}_p^*$, kde p je prvočíslo.

Připomeňme, že nejmenší podgrupa grupy $\mathbf{G}$ obsahující danou množinu $A \subseteq G$ se nazývá *podgrupa generovaná množinou* A a značí se $\langle A \rangle_{\mathbf{G}}$.

Tvrzení 1.2.1. *Nechť $\boldsymbol{G}$ je grupa a $A \neq \emptyset$ je podmnožina G. Pak*

$$\langle A \rangle_{\boldsymbol{G}} = \left\{ a_1^{k_1} \cdot a_2^{k_2} \cdots a_n^{k_n} : \ a_1, \ldots, a_n \in A, \ k_1, \ldots, k_n \in \mathbb{Z} \right\}.$$

Důkaz. Označme

$$S = \left\{ a_1^{k_1} \cdot a_2^{k_2} \cdots a_n^{k_n} : \ a_1, \ldots, a_n \in A, \ k_1, \ldots, k_n \in \mathbb{Z} \right\}.$$

Protože $a_i \in A$, máme také $a_i^{-1} \in \langle A \rangle$, neboli $a_i^{k_i} \in \langle A \rangle$ a tedy i násobení prvků z A náleží $\langle A \rangle$. Proto každý prvek z množiny S lze vygenerovat z množiny A.

Nyní ověříme, že S je podgrupa. Uzavřenost na násobení je zřejmá a z vlastností grup plyne, že $(a_1^{k_1} \cdots a_n^{k_n})^{-1} = a_1^{-k_1} \cdots a_n^{-k_n}$ a konstanta 1 lze napsat jako a^0.

□

Definice. Počet prvků grupy $\mathbf{G}$ se nazývá *řád grupy* $\mathbf{G}$ a značí se $|G|$. Počet prvků grupy $\langle a \rangle_{\mathbf{G}}$, $a \in \mathbf{G}$ se nazývá *řád prvku* a a značí se $|a|$.

Příklad. Grupa $\mathbb{Z}_7^*$ má řád 6. Pro jednotlivé prvky v ní máme: $|1| = 1, \ |2| = 3, \ |3| = 6, \ |4| = 3, \ |5| = 6, \ |6| = 2$.

Tvrzení 1.2.2. *Nechť $\mathbf{G}$ je konečná grupa a $a \in G$. Pak $|a|$ je rovno nejmenšímu kladnému n takovému, že $a^n = 1$.*

Důkaz. Z tvrzení 1.2.1 plyne

$$\langle a \rangle_{\mathbf{G}} = \left\{ a^k | k \in \mathbb{Z} \right\}.$$

Položme $k = qn + r$, kde $q = k \text{ div } n$ a $r = k \mod n$, neboli $0 \leq r < n$. Pokud $a^n = 1$, pro nějaké $n > 0$, pak

$$a^k = a^{nq+r} = (a^n)^q a^r = 1^q a^r = a^r.$$

Tedy $| \langle a \rangle | = n$, kde $n > 0$ a je nejmenší, pro které platí $a^n = 1$. □

Příklad. Grupa $\mathbb{Z}_7^*$ má 6 prvků a to $1, 2, 3, 4, 5, 6$, tedy má tedy řád 6. Pro jednotlivé prvky v ní máme: $|1| = 1, \ |2| = 3, \ |3| = 6, \ |4| = 3, \ |5| = 6, \ |6| = 2$.

Tvrzení 1.2.3. *Nechť $\mathbf{G}$ je grupa a $a \in G$. Pak $|a|$ dělí $|G|$.*

Důkaz. Označme $H = \langle a \rangle_{\mathbf{G}}$. Víme že $H \subseteq G$ a tedy stačí dosadit do Lagrangeovy věty. □

Definice. Grupa $\mathbf{G}$ se nazývá *cyklická*, pokud je generována jedním prvkem. Tedy pokud $\mathbf{G} = \langle a \rangle_{\mathbf{G}}$, kde $a \in G$. Prvek a se nazývá *generátor* grupy $\mathbf{G}$.

Tvrzení 1.2.4. *Nechť $\mathbf{G}$ je cyklická grupa řádu n. Pak $\mathbf{G}$ je izomorfní grupě $\mathbb{Z}_n$.*

Důkaz. Předpokládejme, že $\mathbf{G} = \langle a \rangle$. Dále víme, že $|a| = n$. Definujme

$$f : \mathbb{Z}_n \to \mathbf{G}, \qquad f(k) \mapsto a^k.$$

Ověříme, že f je homomorfismus. Je-li $k + l < n$, pak

$$f(k+l) = a^{k+l \mod n} = a^k a^l = f(k) \cdot f(l).$$

Pokud $k + l > n$ máme

$$f(k+l) = a^{k+l \mod n} = a^{k+l-n} \cdot 1 = a^{k+l-n} a^n = a^{k+l} a^{n-n} = a^{k+l} =$$

$$= a^k a^l = f(k) \cdot f(l).$$

Protože se jedná o prosté zobrazení mezi stejně velkými množinami, pak každému z n prvku množiny $\mathbb{Z}_n$ přiřadí zobrazení f hodnotu. Tyto hodnoty jsou různé, a proto obor hodnot zobrazení f musí mít n prvků, tedy to musí být celá grupa $\mathbf{G}$. Tím jsme dokázali, že je f bijekce. □

Definice. Nejmenší přirozené číslo m takové, že $a^m = 1$, $\forall a \in G$ se nazývá *exponentem* grupy $\mathbf{G}$, pokud takové m existuje a pokud neexistuje říkáme, že exponent $\mathbf{G}$ je nekonečný.

Poznámka. Takový exponent grupy $\mathbf{G}$ podle tvrzení 1.2.3 dělí řád grupy $\mathbf{G}$ a rovná se nejmenšímu společnému násobku všech řádů v grupě $\mathbf{G}$.

Příklad. Exponent Z_7^* je tedy $NSN(3, 6, 2) = 6$

Věta 1.2.5. *Je-li p prvočíslo, pak $\mathbb{Z}_p^*$ je cyklická grupa.*

Důkaz. Nejprve dokážeme, že $p-1$ je exponentem grupy $\mathbb{Z}_p^*$. Pro spor uvažujme, že $n < p - 1$ je také exponent. To znamená, že pro všechny $a \in \mathbb{Z}_p^*$ platí

$$a^n = 1 \mod p.$$

Neboli, že všechny $a \in \mathbb{Z}_p^*$ jsou kořeny polynomu $x^n - 1 \in \mathbb{Z}_p[x]$. Takový polynom má nejvýše n kořenů, protože $\mathbb{Z}_p$ je oborem integrity. Tedy to nejsou všechna $a \in \mathbb{Z}_p^*$ a tak vzniká spor.

Nyní víme, že $p - 1$ je exponent grupy $\mathbb{Z}_p^*$ a abychom dokázali větu, musíme najít prvek (generátor), který je řádu $p - 1$.
Položme $p - 1 = p_1^{l_1} \cdots p_n^{l_n}$ prvočíselný rozklad $p - 1$. Pro každé $i = 1, \ldots, n$ platí, že polynomy

$$x^{\frac{p-1}{p_i}} - 1$$

mají nejvýše $\frac{p-1}{p_i} < p-1$ kořenů, a tedy musí existovat prvky a_i takové, že

$$a_i^{\frac{p-1}{p_i}} \neq 1.$$

Položme

$$b_i = a_i^{\frac{p-1}{p_i^{l_i}}}.$$

Umocněním této rovnice na $p_i^{l_i}$ máme

$$b_i^{p_i^{l_i}} = a_i^{p-1} = 1.$$

Z toho plyne, že $|b_i| \leq p_i^{l_i}$, a nyní chceme dokázat rovnost. Pro spor předpokládejme řád m, kde $0 < m < p_i^{l_i}$. Pak díky Bezoutově rovnosti

$$b_i^{NSD(m,p_i^{l_i})} = b_i^{um+vp_i^{l_i}} = b_i^{um} \cdot b_i^{vp_i^{l_i}} = (b_i^m)^u \cdot (b_i^{p_i^{l_i}})^v = 1 \cdot 1 = 1.$$

Označíme-li ale $NSD(m, p_i^{l_i}) = p^k$, kde $0 \leq k < l_i$ je

$$1 = b_i^{p^k} = (a_i^{\frac{p-1}{p_i^{l_i}}})^{p^k} = a_i^{\frac{p-1}{p_i^{l_i-k}}} \neq 1.$$

Tedy dostaneme spor, protože $\frac{p-1}{p_i^{l_i-k}} < \frac{p-1}{p_i}$.

V další části dokážeme, že pokud prvek a je řádu u a b je řádu v, kde $u,\ v$ jsou nesoudělná, pak řád ab je rovný uv. Protože

$$(ab)^{uv} = ((a)^u)^v \cdot ((b)^v)^u = 1 \cdot 1 = 1,$$

pak $|ab| \leq uv$. Nechť tedy máme nějaké $r < uv$ takové, že $|ab| = r$. Číslo $r|uv$ a můžeme si jej tedy psát ve tvaru $r = u'v'$, kde $u'|u$ a $v'|v$. Bez újmy na obecnosti máme $u' < u$, $r|u'v$ a platí

$$1 = (ab)^{u'v} = a^{u'v}(b^v)^{u'} = a^{u'v} \cdot 1 = a^{u'v}.$$

Tedy u musí dělit $u'v$, ale to nelze, protože u je nesoudělné s v a $u' < u$. Dokázali jsme druhou část a nyní položme

$$a = b_1 \cdots b_n.$$

Podle předchozího máme

$$|a| = |b_1| \cdots |b_n| = p_1^{l_1} \cdots p_n^{l_n} = p - 1.$$

Našli jsme tedy prvek řádu $p-1$, neboli generátor grupy $\mathbb{Z}_p^*$.

□

Poznámka. Pomocí této věty a tvrzení 1.2.4 dostáváme, že $\mathbb{Z}_p^*$ je izomorfní $\mathbb{Z}_{p-1}$.

1.3 Diskrétní logaritmus

Bezpečnost většiny asymetrických šifrovacích metod, mezi které patří i digitální podpisy, je založena na obtížnosti výpočtu diskrétních logaritmů. V následujícím textu budeme opět předpokládat, že **G** je konečná multiplikativní grupa, tj. $\mathbf{G} = (G, \cdot, ^{-1}, 1)$.

Definice. Mějme **G** cyklickou grupu a $a \in G$ její generátor. Pak pro každé $b \in G$ existuje právě jeden exponent $k \in \{0, \ldots, |G| - 1\}$ takový, že

$$a^k = b.$$

Toto číslo se nazývá *diskrétní logaritmus* prvku b o základu a v grupě G a značí se $\log_a b$.

Poznámka. V grupě $\mathbb{Z}_p^*$, kde p je prvočíslo a $a \in \mathbb{Z}_p^*$ je její generátor, je $\log_a b$ rovno takovému $k \in \{0, \ldots, p-2\}$, že

$$a^k \mod p = b.$$

Zatím není známý žádný obecný algoritmus pracující v polynomiálním čase vzhledem k p, který by našel takové k.

Příklad. V $\mathbb{Z}_7^*$ je $\log_3 4 = 4$, protože $3^4 = 4 \mod 7$.

Rekordy ve výpočtu diskrétního logaritmu

8.4.2004 - Společnost Certicom vyhlásila soutěž na nalezení řešení několika kryptografických úloh, mezi které patřil i diskrétní logaritmus. Výpočet diskrétního logaritmu proběhl použitím metody na bázi eliptických křivek a výpočet trval 17 měsíců. (Klíma, Vyřešena úloha diskrétního logaritmu ECC2-109, Security - News, 2004 [6])

5.2.2007 - Thorsten Kleinjung (Mathematical Institute of the University of Bonn) vypočítal diskrétní logaritmus použitím metody číselného síta. Prvočíslo p mělo 530 bitů, 160 dekadických číslic. (Kleinjung, Discrete logarithms in GF(p), 2007 [7])

8.7.2009 - Byl vyřešen problém nalezení eliptického diskrétního logaritmu pro eliptickou křivku v prvočíselném 112-bitovém poli. Bylo použito více než 200 konzolí PlayStation 3 a trval téměř 6 měsíců. Na výpočtu spolupracovali Joppe W. Bos, Marcelo E. Kaihara1, Thorsten Kleinjung, Arjen K. Lenstra a Peter L. Montgomery. (Bos and Kaihara, PlayStation 3 computing breaks 2^{60} barrier 112-bit prime ECDLP solved, LACAL, 2009 [8])

1.3.1 Některé algoritmy řešící problém diskrétního logaritmu

Bezpečnost digitálních podpisů, kterými se budeme dále zabývat, je založena na složitosti výpočtu diskrétního logaritmu. Proto si ukážeme některé známe algoritmy na výpočet takového diskrétního logaritmu. Budeme zde čerpat z [3] a [5].

Baby-step giant-step algoritmus

Algoritmus Baby-step giant-step je použitelný pro jakoukoliv cyklickou grupu $\mathbf{G}$. Vezmeme $m = \lceil \sqrt{n} \rceil$, kde n je řád a. Má-li platit $a^k = b$, pak k můžeme zapsat

$$k = im + j,$$

pro $0 \leq i, j < m$. Tedy

$$b = a^k = a^{im} a^j$$

a po úpravě dostaneme

$$b(a^{-m})^i = a^j.$$

V algoritmu si vypočteme pravou stranu pro každé j, levou stranu pro každé i a pro ty i, j, která budou splňovat uvedenou rovnici vypočteme k. Složitost tohoto algoritmu je $O(\sqrt{n})$.

Pollardův rho algoritmus

Tento algoritmus má stejnou asymptotickou složitost jako Baby-step giant-step algoritmus $O(\sqrt{n})$, ale nepotřebuje tolik paměťového místa na mezivýsledky. Používá se taktéž pro jakoukoliv cyklickou grupu $\mathbf{G}$. Chceme opět, aby platilo $\log_a b = k$. Funguje tak, že grupu $\mathbf{G}$, ve které hledáme k, si rozdělíme na tři podmnožiny $S_1,\ S_2,\ S_3$ podobné velikosti. Dále definujme posloupnost $x_0, x_1, x_2, \ldots$, kde $x_0 = 1$ a

$$x_{i+1} = f(x_i) = \begin{cases} bx_i, & x_i \in S_1, \\ x_i^2, & x_i \in S_2, \\ ax_i, & x_i \in S_3, \end{cases}$$

pro $i \geq 0$. Pomocí této posloupnosti prvků definuje dvě posloupnosti celých čísel $\alpha_0, \alpha_1, \ldots$ a $\beta_0, \beta_1, \ldots$, splňující $x_i = a^{\alpha_i} b^{\beta_i}$ pro každé $i \geq 0$, kde $\alpha_0 = 0,\ \beta_0 = 0,$

a dále

$$\alpha_{i+1} = \begin{cases} \alpha_i, & x_i \in S_1, \\ 2\alpha_i \pmod{n}, & x_i \in S_2, \\ \alpha_i + 1 \pmod{n}, & x_i \in S_3, \end{cases}$$

a

$$\beta_{i+1} = \begin{cases} \beta_i + 1 \pmod{n}, & x_i \in S_1, \\ 2\beta_i \pmod{n}, & x_i \in S_2, \\ \beta_i, & x_i \in S_3. \end{cases}$$

Pomocí Floydova algoritmu, který je popsán například zde [9], pak můžeme nalézt x_i a x_{2i} taková, že $x_i = x_{2i}$, a tedy

$$a^{\alpha_i} b^{\beta_i} = a^{\alpha_{2i}} b^{\beta_{2i}} \;\Rightarrow\; b^{\beta_i - \beta_{2i}} = a^{\alpha_{2i} - \alpha_i}.$$

Zlogaritmujeme-li tuto rovnici logaritmem o základu α, dostaneme

$$(\beta_i - \beta_{2i}) \log_a b = (\alpha_{2i} - \alpha_i) \pmod{n}.$$

Vezmeme-li $\beta_i \neq \beta_{2i} \mod n$ (pravděpodobnost rovnosti těchto prvků je zanedbatelná), pak tato rovnice je řešením $\log_a b$.

Pohligův-Hellmanův algoritmus

Pohligův-Hellmanův algoritmus se nejvíce používá pro cyklické grupy $\mathbf{G}$ řádu n, kde n má relativně malé prvočíselné dělitele. Na začátku algoritmu si najdeme prvočíselný rozklad čísla n. Mějme tedy rozklad

$$n = p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m},\ m, e \in \mathbb{Z}.$$

Má-li platit $k = \log_a b$, pak pomocí Čínské věty o zbytcích vypočteme $k \pmod{n}$ z m rovnic

$$\begin{aligned} k_1 &= k \pmod{p}^{e_1}, \\ &\vdots \\ k_m &= k \pmod{p}^{e_m}. \end{aligned}$$

Každé $k_i \in \mathbb{Z}$, kde $i \in \mathbb{Z}$, lze vyjádřit pomocí p_i-adické reprezentace, tj.

$$k_i = l_0 + l_1 p_i + l_2 p_i^2 + \ldots l_{e_i - 1} p_i^{e_i - 1},\ 0 \leq l_j \leq p_i - 1.$$

Všechna l_j, kde $j = 0, 1, \ldots, e_i - 1$, vypočítáme následovně. Položme

$$\alpha = a^{\frac{n}{p_i}}, \qquad \gamma = 1 \qquad \text{a} \qquad l_{-1} = 0.$$

a dále vypočítáme

$$\gamma = \gamma\alpha^{l_{j-1}} p_i^{j-1} \qquad \text{a} \qquad \beta = (b\gamma^{-1})^{\frac{n}{p_i^{j+1}}}.$$

Pak už jen pomocí Baby-step giant-step vypočítáme

$$log_\alpha \beta = l_j,\ 0 \leq j \leq e_i - 1.$$

Složitost Pohligova-Hellmanova algoritmu je $O(\sum_{i=1}^{m} e_i(\log n + \sqrt{p_i}))$.

Algoritmus indexového výpočtu

Algoritmus pod anglickým názvem „Index-calculus algoritmus“ je zřejmě nejúčinnější metodou pro výpočet diskrétního logaritmu. Používá se však jen pro některé cyklické grupy, nejvíce pro $\mathbb{Z}_p^*$. Použití této metody ukážeme na této grupě. Nejprve si zvolíme tzv. faktorizační bázi $B = \{p_1, p_2 \dots p_t\},\ t \in \mathbb{Z}$, která obsahuje prvních t prvočísel. Z hlediska výpočetní efektivity by bylo třeba zvolit co nejmenší hodnotu t, ale k tomu, abychom našli řešení následující soustavy lineárních rovnic, je třeba volit co nejvyšší hodnotu t. Proto je třeba volit kompromis. Dále zvolíme $x_j \in \mathbb{Z}_{p-1}$, pro $1 \leq j \leq s$, kde s je o něco větší než t, a snažíme se rozložit a^{x_j} na prvočíselný rozklad nad bází B

$$a^{x_j} = \prod_{i=1}^{t} p_i^{e_{i,j}},\ e_i \geq 0.$$

Pokud se nám to podaří, zlogaritmujeme tuto rovnici logaritmem o základě a a dostaneme

$$x_j = \sum_{i=1}^{t} e_{i,j} \log_a p_i \pmod{p-1}.$$

Nyní máme $s \geq t$ rovnic a proto s velkou pravděpodobností tato soustava rovnic bude mít právě jedno řešení. Tedy nalezneme hodnoty $\log_a p_i,\ 1 \leq i \leq t$.

V další části si zvolíme takové $x \in \mathbb{Z}_{p-1}$, aby se nám podařilo rozložit ba^x na prvočíselný rozklad nad bází B

$$ba^x = \prod_{i=1}^{t} p_i^{d_i},\ d_i \geq 0.$$

Následně zlogaritmujeme tuto rovnici opět logaritmem o základě a a dostaneme

$$\log_a b = k = \sum_{i=1}^{t} d_i \log_a p_i - x \pmod{n}.$$

Složitost indexového výpočtu je $e^{(c+O(1))(\log p)^{\frac{1}{3}}(\log\log p)^{\frac{2}{3}}},\ 0 < c < 1,587$.

Algoritmus indexového výpočtu má několik různých podob. Nejznámější modifikace je metoda číselného síta.

2. DSA - Digital Signature Algorithm

Dne 19.8.1991 navrhla americká organizace NIST nový standard DSS (Digital Signature Standard), který je veden pod označením FIPS-186. Návrh obsahoval následující hlavní cíle.

- Použití veřejného klíče k ověřování integrity dat a autentizace podepisujícího. Možnost ověření pravosti pomocí třetí osoby.
- Jednoduchá implementace v hardware i v software k použití v elektronické poště, k elektronickému převodu finančních prostředků, k distribuci dat, či softwaru a dalších operací, které vyžadují zajištění integrity a ověření původu. V návrhu padla zmínka i o použití v čipových kartách.
- Snadný export mimo USA.

V roce 1994 došlo k přijetí návrhu a byl standardizován ve FIBS PUB 186 pro práci s informacemi bez označení stupně utajení. Novější verze standardu FIBS PUB 186-2 stanovuje, že jako hašovací funkce se využívá SHA1. V roce 2000 pak byl do standardu zahrnut i algoritmus RSA (ve stejné době vypršel i patent na RSA) odkazem na standard ANSI X9.31. Nejnovější verze je FIBS PUB 186-3 vydaná v červnu 2009 [10], která zavádí hašovací funkci SHA-2. Dále zavadí čtyři možnosti velikosti prvočísel p a q v bitech:

- $p = 1024, q = 160$,
- $p = 2048, q = 224$,
- $p = 2048, q = 256$,
- $p = 3072, q = 256$.

Ve standardu jsou popsány dvě metody na výběr takových prvočísel. První metoda je založena na náhodném výběru dvou celých čísel p a q. Poté se pomocí pravděpodobnostního algoritmu ověří, zda jsou vybraná čísla prvočísla. Druhá metoda vytváří celá čísla pomocí malých čísel tak, aby vytvořená čísla byla s jistotou prvočísla.

David W. Kravitz nechal DSA patentovat patentem 5231668 a NIST tento patent nechal k celosvětovému používaní bez poplatků.

V České republice byl vydán zákon 227/2000 Sb. o elektronickém podpisu dne 29.6.2000 [11]. Zákon byl dále novelizován, naposledy pak zákonem č. 281/2009 Sb. a zákonem č. 424/2010 Sb. „*Tento zákon upravuje v souladu s právem Evropských společenství používání elektronického podpisu, elektronické značky, poskytování certifikačních služeb a souvisejících služeb poskytovateli usazenými na území České republiky, kontrolu povinností stanovených tímto zákonem a sankce za porušení povinností stanovených tímto zákonem.*“ Definuje základní pojmy a stanovuje požadavky kladené na elektronický podpis. V zákoně byly zavedeny nové pojmy jako „kvalifikované časové razítko“ a „elektronické značky“. Zákon stanovuje všechny povinnosti osob a služeb vztahujících se k elektronickému podpisu a také přidává povinnost vést a zveřejňovat seznam důvěryhodných certifikačních služeb a uznávat kvalifikované certifikáty vydané v ostatních členských státech EU a Švýcarské konfederaci.

Dále v této i další kapitole budeme čerpat z Mollina [12]

2.1 Algoritmus

Generování klíčů

1. Náhodně si zvolíme dvě prvočísla p a q taková, že $q|p-1$.
2. Vybereme číslo g takové, že $1 < g < p$ a zároveň jeho řád byl q. To znamená, aby $g^q = 1 \pmod{p}$. Takové číslo nejčastěji najdeme pomocí dosazování do $g = a^{\frac{p-1}{q}} \pmod{p}$, kde $1 < a < p-1$, dokud výsledek nebude různý od jedné. Nejčastěji se používá $a = 2$.
3. Náhodně vybereme $x \in \langle 1, q-1 \rangle$ a vypočítáme $y = g^x \pmod{p}$.
4. Nyní už můžeme zveřejnit (p, q, g, y) jako veřejný klíč. Ponecháme si x jako soukromý klíč.

Podpis zprávy m

1. Vybereme si hašovací funkci h.
2. Zvolíme si náhodně jednorázovou hodnotu $k \in \mathbb{Z}_q^*$ tzv. nonce, která by se neměla opakovat.
3. Vypočítáme

$$r = (g^k \pmod{p}) \pmod{q} \qquad \text{a} \qquad s = k^{-1}(h(m)+xr) \pmod{q}.$$

4. Podpis (r, s) pošleme druhé straně.

Ověření podpisu

1. Vypočítáme $w = s^{-1} \pmod{q}$.
2. Pomocí veřejných klíčů dále počítáme $u_1 = h(m)w \pmod{q}$ a $u_2 = rw \pmod{q}$.
3. Pak už jen vypočteme $v = ((g^{u_1}y^{u_2}) \pmod{p}) \pmod{q}$.
4. Podpis je platný, pokud $r = v$.

2.2 Správnost algoritmu

Tvrzení 2.2.1. *Nechť p a q jsou prvočísla taková, že $q|p-1$, a je kladné celé číslo menší než p a $g \equiv a^{\frac{p-1}{q}} \pmod{p}$. Pak $g^q \equiv 1 \pmod{p}$, a pokud $m \pmod{q} = n \pmod{q}$, pak $g^m \pmod{p} = g^n \pmod{p}$.*

Důkaz. Víme, že

$$g^q \pmod{p} = (a^{\frac{p-1}{q}} \pmod{p})^q \pmod{p} = a^{p-1} \pmod{p} = 1.$$

Nyní nechť $m \pmod{q} = n \pmod{q}$, to znamená $m = n + kq$ pro nějaké $k \in \mathbb{Z}$. Pak

$$g^m \pmod{p} = g^{n+kq} \pmod{p} = g^n g^{kq} \pmod{p} =$$
$$= ((g^n \pmod{p})(g^q \pmod{p})^k) \pmod{p} = g^n \pmod{p}.$$

□

Věta 2.2.2. *Podpis je platný, pokud $r = v$.*

Důkaz. Z algoritmu na vytvoření podpisu $s = k^{-1}(h(m) + xr) \pmod{q}$ máme

$$k \equiv h(m)s^{-1} + xrs^{-1} \equiv h(m)w + xrw \pmod{q}.$$

Protože g má řád q, platí

$$g^k \equiv g^{h(m)w+xrw} \equiv g^{h(m)w} g^{xrw} \equiv g^{h(m)w} y^{rw} \equiv g^{u_1} y^{u_2} \pmod{p}.$$

Z toho tedy plyne

$$r = (g^k \pmod{p}) \pmod{q} = ((g^{u_1}y^{u_2}) \pmod{p}) \pmod{q} = v.$$

□

2.3 Složitost algoritmu

Nyní se podívejme na časovou složitost, aby jsme věděli, jak je náročný výpočet algoritmu. Označme $n = l(p)$ jako počet cifer čísla p a předpokládejme, že číslo q je podobně velké.

Při generování klíče ve 2. a 3. kroku používá výpočet mocnění v tělese $\mathbb{Z}_p^*$, které má složitost $O(n^2)$. Tedy generování klíčů má složitost

$$2 \cdot O(n^2) = O(n^2).$$

Samotný podpis vyžaduje výpočet haše, který má složitost $O(n)$. Ve 3. kroku výpočet v tělese $\mathbb{Z}_q^*$ obnáší jedno mocnění, násobení a výpočet inverze se složitostí $O(n^2)$ a součet, který má složitost $O(n)$. Tedy podepisování má složitost

$$3 \cdot O(n^2) + O(n) = O(n^2).$$

Ověřování se počítá opět v tělese $\mathbb{Z}_q^*$. V 1. kroku výpočtu máme výpočet inverze, v 2. kroku výpočet haše a dvě násobení a ve 3. kroku už jen dvě mocnění. Tedy ověřování má složitost

$$5 \cdot O(n^2) + O(n) = O(n^2).$$

3. Schnorr

Mezi lety 1989 - 1991 německý profesor matematiky Claus-Peter Schnorr nechal patentovat svůj návrh na digitální podpis. Byl to patent 4995082, který vypršel v únoru roku 2008. Schnorr upozorňoval, že byl jeho patent porušen vytvořením DSA, jelikož jeho patent DSA pokrývá také.

3.1 Algoritmus

Generování klíčů

1. Náhodně si zvolíme dvě prvočísla p a q taková, že $q|p-1$.
2. Vybereme číslo g takové, že $1 < g < p$ a zároveň jeho řád byl q. To znamená, aby $g^q = 1 \pmod{p}$. Takové číslo nejčastěji najdeme pomocí dosazování do $g = a^{\frac{p-1}{q}} \pmod{p}$, kde $1 < a < p-1$, dokud výsledek nebude různý od jedné. Nejčastěji se používá $a = 2$.
3. Náhodně vybereme $x \in \langle 1, q-1 \rangle$ a vypočítáme $y = g^x \pmod{p}$.
4. Nyní už můžeme zveřejnit (p, q, g, y) jako veřejný klíč a ponecháme si x jako soukromý klíč.

Podpis zprávy m

1. Vybereme si hašovací funkci h.
2. Zvolíme si náhodně jednorázovou hodnotu $k \in \mathbb{Z}_q^*$ tzv. nonce, která by se neměla opakovat.
3. Vypočítáme $e = (g^k \pmod{p}) \pmod{q}$,
   $r = h(m||^1 e)$ a $s = xr + k \pmod{q}$
4. Podpis (r, s) pošleme druhé straně.

Ověření podpisu

Podpis je platný jedině, pokud

$$h(m||g^s y^{-r} \pmod{p}) = r.$$

[1] symbolem || značíme zřetězení neboli spojení dvou čísel.

3.2 Správnost algoritmu

Budeme postupovat podobně jako u DSA.

Věta 3.2.1. *Podpis je platný, pokud $r = v$.*

Důkaz. Z algoritmu na vytvoření podpisu $s = xr + k \pmod q$ máme

$$k \equiv s - xr \pmod q.$$

Protože g má řád q, platí

$$g^k \equiv g^{s-xr} \equiv g^s g^{-xr} \equiv g^s (g^x)^{-r} \equiv g^s y^{-r} \pmod p.$$

Z toho tedy plyne

$$r = h(m||g^k \pmod p)) = h(m||g^s y^{-r}) \pmod p = v.$$

□

3.3 Složitost algoritmu

Opět označme $n = l(p)$ jako počet cifer čísla p a předpokládejme, že číslo q je podobně velké.

Generování klíče funguje stejně jako v předchozí kapitole. Tedy jeho složitost je

$$2 \cdot O(n^2) = O(n^2).$$

Samotný podpis v tělese $\mathbb{Z}_q^*$ vyžaduje ve 3. kroku jedno mocnění, zřetězení se složitostí $O(n)$,výpočet haše, násobení a součet. Tedy podepisování má složitost

$$2 \cdot O(n^2) + 3 \cdot O(n) = O(n^2).$$

Ověřování probíhá opět v tělese $\mathbb{Z}_q^*$. Algoritmus obnáší výpočet inverze, dvě mocnění, zřetězení a výpočet haše. Tedy ověřování má složitost

$$3 \cdot O(n^2) + 2 \cdot O(n) = O(n^2).$$

Z toho plyne, že časová složitost je shodná jako u DSA.

4. Některé útoky na DSA

Nyní už umíme podepisovat zprávy pomocí digitálního podpisu. Musíme si však uvědomit, že existují útočníci, který by chtěli podpis padělat. To znamená vytvořit podpis, který bude nerozlišitelný od našeho podpisu. Toho dosáhne, pokud se pokusí prolomit schéma digitálního podpisu, který používáme. Může se jednat o různé typy prolomení.

- *Úplné prolomení* - útočník je schopen vypočítat soukromý klíč podepisujícího ze znalosti veřejného klíče.
- *Univerzální podvržení* - útočník nalezne díky znalosti veřejného klíče efektivní algoritmus, který dokáže vytvořit platný podpis ekvivalentní tomu, co vytváří podepisující.
- *Selektivní podvržení* - útočník je schopný vytvořit platný podpis, ale pouze pro konkrétní zprávu.
- *Existenční podvržení* - útočník je chopen vytvořit platný podpis zprávy, ale nemá žádnou kontrolu nad tím, jak zpráva vypadá.

Těchto prolomení útočník může dosáhnout při úspěšném útoku na schéma. Základní útoky jsou:

1. útok se znalostí klíče - útočník zná pouze veřejný klíč podepisujícího a snaží se získat jeho soukromý klíč,
2. útok se znalostí zprávy - útočník má veřejný klíč a také má některé podepsané zprávy, ale nemůže změnit jejich obsah. Podepsanou zprávu může dostat následovně:
   (a) podepisující podepíše útočníkovu podstrčenou zprávu, jejíž obsah útočník sám vytvoří,
   (b) útočník může opakovaně předkládat podepisujícímu podstrčené zprávy, které závisí na předchozí zprávě.

V kapitole čerpáme z Hlaváčových skript [13], Schneiera [2], Caoa [15] a Markowitche [14].

4.1 Útoky na soukromý klíč

1. **Zná-li útočník hodnotu** k, pak může získat soukromý klíč x tak, že vynásobí-li rovnici

$$s = k^{-1}(h(m) + xr) \pmod{q}$$

hodnotou k, pak díky vlastnosti $k^{-1}k \equiv 1 \mod q$ dostane

$$sk = (h(m) + xr) \pmod{q}$$

a odtud už jednoduchou úpravou

$$x = r^{-1}(sk - h(m)) \pmod{q}.$$

2. **Při generování** útočník zná

$$y = g^x \pmod{p}.$$

Pomocí některých z výše uvedených algoritmů na diskrétní logaritmus může útočník zkusit najít takové x, aby rovnice platila.

4.2 Útoky na jednorázovou hodnotu

1. Jestliže se **opakuje hodnota** k, útočník ví, že podpisy (r_1, s_1) a (r_2, s_2) zpráv $m_1,\ m_2$ byly vygenerované stejnou hodnotou k, protože $r_1 = r_2$. Tedy má k dispozici dvě rovnosti

$$s_1 = k^{-1}(h(m_1) + xr_1) \pmod{q},$$

$$s_2 = k^{-1}(h(m_2) + xr_1) \pmod{q}.$$

Vynásobením obou rovnic hodnotou k dostane

$$s_1k = (h(m_1) + xr_1) \pmod{q},$$

$$s_2k = (h(m_2) + xr_1) \pmod{q}.$$

Dále tyto rovnice pouze odečte od sebe a získá

$$k(s_1 - s_2) = h(m_1) - h(m_2) \pmod{q}$$

$$k = (s_1 - s_2)^{-1}(h(m_1) - h(m_2)) \pmod{q}.$$

Nyní může dostat soukromý klíč z již uvedené metody.

2. **Při ověřování** máme

$$g^{u_1}y^{u_2} \pmod p = g^{h(m)s^{-1} \pmod q}g^{x(rs^{-1}) \pmod q} \pmod p.$$

Z rovnice $s = k^{-1}(h(m)+xr) \mod q$ útočník vidí, že celý exponent je rovný hodnotě k. Tedy útočník umí vypočítat $g^k \mod p$ pouze pomocí veřejných parametru DSA a podpisu.

Hodnotu k pak může zkusit vypočítat pomocí jednoho z algoritmů na výpočet diskrétního logaritmu. Některé speciální případy diskrétního logaritmického problému se dají řešit Pohligovým - Hellmanovým algoritmem, který vyžaduje, aby $p-1$ byla B-hladká hodnota pro nějaké B „malé“ tzn. $p-1 = \prod_i q_i$, kde $\forall q_i < B$.

4.3 Útoky na zprávu

1. Pomocí **ověřování** se útočník může dozvědět $h(m)$, pokud zná veřejné klíče a podpis. Ví, že

$$r = g^{h(m)s^{-1}}y^{rs^{-1}},$$

a po úpravě dostane

$$g^{h(m)} = r^s y^{-r}.$$

Pravou stranu útočník vypočte a může zjistit $h(m)$, jestliže umí řešit diskrétní logaritmus.

2. **Útok se znalostí zprávy**
V tomto případě útočník chce podepsat svoji zprávu m naším podpisem. Nastíníme si, jak by takový útok měl vypadat. Nejprve si útočník vypočte $h(m)$. Dále ví, že při ověřování je

$$r = ((g^{h(m)s^{-1}}y^{rs^{-1}}) \pmod p) \pmod q.$$

Předpokládá tedy, že

$$r = ((g^a y^b) \pmod p) \pmod q$$

pro zatím neznámá $a,\ b$. Tedy platí

$$(g^a y^b)^s = g^{h(m)}y^{(g^a y^b \pmod p)} \pmod p.$$

Porovná mocniny a vidí, že

$$as = h(m) \pmod q,$$

$$bs = (g^a y^b \pmod p) \pmod q.$$

Z tohoto už lehce dostane

$$h(m) = ab^{-1}(g^a y^b \pmod p) \pmod q. \qquad (*)$$

K dané zprávě stačí útočníkovi, aby našel $a, b \in \mathbb{Z}_q^*$, která splňují $(*)$. Pak mu již zbývá vypočítat

$$r = ((g^a y^b) \pmod p) \pmod q,$$

$$s = rb^{-1} \pmod q.$$

Výsledný podpis zprávy m pak je (r, s).

4.4 Porovnání se Schnorrovým algoritmem

Podpisové schéma DSA a Schnorra je velice podobné. Jsou to sice jen dva z tisíce podpisových algoritmů založené na problému výpočtu diskrétního logaritmu, ale jsou to ti nejznámější zástupci.

Generování klíče probíhá stejně a útočník by se mohl pokusit získat soukromý klíč u Schnorrova algoritmu stejně jako u DSA pomocí některého z algoritmů na výpočet diskrétního logaritmu.

Při podepisování zprávy m se zvolí jednorázová hodnota k menší než q a vypočítá se

$$r = (g^k \pmod p) \pmod q.$$

Obecně se pak počítá rovnice tvaru

$$ak = b + cx \pmod q.$$

Koeficienty $a,\ b,\ c$ mohou být různé variace $h(m),\ r,\ s$ nebo 1. U DSA je vstup do hašovací funkce pouze zpráva m. To není tak bezpečné, protože pokud útočník najde kolizi, může pak použít útok se znalostí zprávy.

U ověřování se ve Schnorrově algoritmu do hašovací funkce vkládá nejen zpráva, ale i podpis samotný. Této vlastnosti se říká sebe-zpětná vazba. Takže útočník by musel najít obsah r pro daná m, p, g a daný haš takový, že platí

$$h(m||g^s y^{-r} \pmod p) = r.$$

Podle požadavků kladených na hašovací funkce by měl být tento výpočet velmi obtížný, i když je známo, že SHA-1 je náchylný ke kolizi. Ve schématu je výsledný podpisový údaj r pevně svázán s danou zprávou m a druhým podpisovým údajem s. To znamená, že vstup hašovací funkce by měl být generován synchronně s výstupem. Útočník tedy ani nemůže výše uvedenou rovnici rozložit na jiné výpočetní úkony, proto nelze útok na zprávu se Schnorrovým podpisem vykonat. U DSA takový vztah mezi $r,\ s$ a m není, jelikož zprávu m zahrnuje do podpisového údaje s, a tak se nedá útočníkovi zabránit, aby zahájil útok.

Při útoku na jednorázovou hodnotu ve Schnorrově algoritmu je to podobné jako v předchozího případě. Útočník by musel pro daná m, g, p a q opět najít r takové, že

$$r = h(m||(g^k \pmod p) \pmod q),$$

a aby následně platilo

$$s = xr + k \pmod q.$$

Zde je tedy samotný podpis (r, s) pevně svázán s jednorázovou hodnotou k a tak, podobně jako u ověřování, je útok na jednorázovou hodnotu nemožný.

Z uvedených informací by se dalo usuzovat, že Schnorrův podpis je bezpečnější a odolnější vůči útokům.

5. Analýza útoku s jednorázovou hodnotou

V této kapitole se zaměříme na útok, který není v učebnicích diskutován. Nejprve se zamyslíme nad tím, jestli vůbec existují dvě jednorázové hodnoty k a k' takové, že

$$(g^k \mod p) \mod q = (g^{k'} \mod p) \mod q.$$

Ukážeme si to na příkladě.

Příklad. Nechť máme $p = 23,\ q = 11,\ g = 4$. Pak máme

$k = 1$ $\quad 4^1 = (4 \mod 23) \mod 11 = 4,$
$k = 2$ $\quad 4^2 = (16 \mod 23) \mod 11 = 5,$
$k = 3$ $\quad 4^3 = (64 \mod 23) \mod 11 = 7,$
$k = 4$ $\quad 4^4 = (256 \mod 23) \mod 11 = 3,$
$k = 5$ $\quad 4^5 = (1024 \mod 23) \mod 11 = 1,$
$k = 6$ $\quad 4^6 = (4096 \mod 23) \mod 11 = 2,$
$k = 7$ $\quad 4^7 = (16384 \mod 23) \mod 11 = 8,$
$k = 8$ $\quad 4^8 = (65536 \mod 23) \mod 11 = 9,$
$k = 9$ $\quad 4^9 = (262144 \mod 23) \mod 11 = 2,$
$k = 10$ $\quad 4^{10} = (1048576 \mod 23) \mod 11 = 6,$

Všimneme si, že tedy pro dvě čísla k je výsledek stejný.

Tedy může nastat určitá kolize, kterou by eventuálně mohl útočník nalézt, a tak si vytvořit stejný platný podpis. Podívejme se nyní, s jakou pravděpodobností pro dané k může takovou kolizi najít.

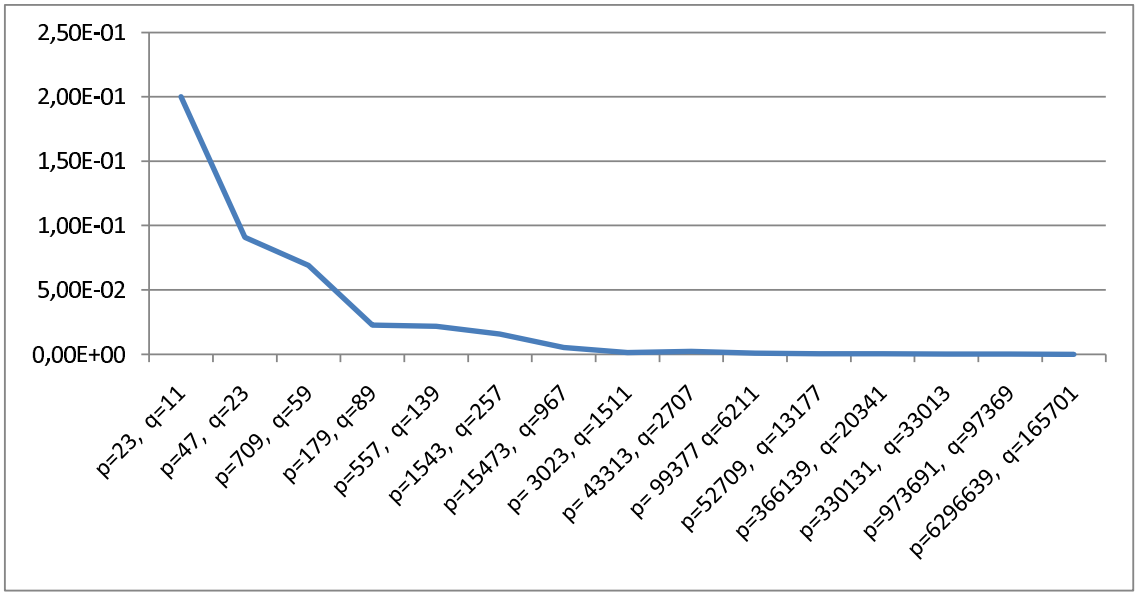
Z příkladu je patrné, že pokud si pro podpis zvolíme $k = 6$ nebo $k = 9$, pak s pravděpodobností $\frac{1}{5}$ nalezne útočník k, aby podpis zůstal stejný. Při výběru jiného k je pravděpodobnost $\frac{1}{10}$. Nyní se podíváme na další jeden příklad pro větší prvočísla p a q.

Příklad. Nechť máme zadáno $p = 47,\ q = 23,\ g = 4$. Po výpočtu jako v předchozím příkladě nám vyjde, že pokud si zvolíme k z množiny čísel

$$\{1, 5, 9, 12, 17, 18\},$$

pak s pravděpodobností $\frac{1}{11}$ útočník nalezne takové k, aby podpis zůstal stejný. Pro ostatní k je pravděpodobnost $\frac{1}{22}$.

Vidíme tedy, že se pravděpodobnost najití jiného k snížila. Mohli bychom se domnívat, že pro hodně velké prvočísla p a q bude pravděpodobnost téměř nulová. Provedla jsem měření právě pro takové vyšší hodnoty, které jsou zaznamenány na následujícím grafu, a také jsem zjistila, že počet takových k, pro které rovnice vychází stejně, není příliš mnoho. Maximálně jich bylo totiž jen pět.



Obrázek 5.1: pravděpodobnost kolize

Graf jsem vytvořila tak, že pro prvočísla p a q, zapsaná na horizontální ose, jsem si vypočetla všechny možnosti jako v uvedených příkladech. Na vertikální ose je pak maximální pravděpodobnost, s jakou útočník může nalézt k a zároveň bude platit rovnice výše uvedená. Z grafu vyplývá, že naše domněnka je správná, a tak si myslím, že v praxi tento útok pro opravdu velká prvočísla nehrozí.

6. Jak to všechno začalo

Všem kryptografickým systémům dlouho chyběl mechanismus, který by zajišťoval bezpečnou distribuci tajných klíčů. Byla známá pouze symetrická kryptografie, která umožňovala útočníkovi ze znalosti šifrovacího klíče získat dešifrovací klíč. V roce 1970 Whitfield Diffie a Martin Hellman představili ve své knize „New directions in cryptography“ metodu, jak se dva komunikující, kteří se nikdy neviděli, dohodnou na sdíleném tajném klíči přes nezabezpečený kanál. Používají se v této metodě dva klíče. Jeden pro šifrování, který může být veřejný, a druhý pro dešifrování, který je tajný. Princip o dohodě klíče je následující:

1. A zveřejní prvočíslo p, konečnou grupu $\mathbb{Z}_p^*$ a generátor g této grupy,
2. A si zvolí svůj soukromý klíč a a spočítá svůj veřejný klíč
$$A = g^a \pmod{p},$$
3. B si zvolí svůj soukromý klíč b a spočítá svůj veřejný klíč
$$B = g^b \pmod{p},$$
4. A spočítá
$$s \equiv B^a \pmod{p},$$
5. B spočítá
$$s' \equiv A^b \pmod{p}.$$

Čísla s a s' jsou stejná, neboť

$$s = B^a \equiv (g^b)^a \equiv g^{ba} \equiv (g^a)^b \equiv= A^b \equiv s' \pmod{p}.$$

Tedy číslo s je dohodnutý klíč. Toto číslo by neměl nikdo zjistit, protože jde o problém diskrétního logaritmu.

Zanedlouho poté Ronald Rivest, Adi Shamir a Len Adleman vytvořili algoritmus RSA, který mohl být používán k tvorbě digitálního podpisu. Bezpečnost RSA je postavena na předpokladu, že rozložit číslo N na součin prvočísel p a q (faktorizace) je velmi obtížná úloha, neboť doposud není znám žádný algoritmus, který by tuto úlohu řešil v polynomiálním čase vůči velikosti N v binárním zápisu. Postup RSA je následující:

- *Generování klíče*
  Náhodně si vybereme dvě prvočísla p a q podobné délky a vypočítáme

$$N = pq,$$

$$\varphi(N) = (p-1) \cdot (q-1).$$

  Pak si zvolíme náhodně $e \in \mathbb{N}$ (šifrovací exponent) takové, že $1 < e < \varphi(N)$ a zároveň $NSD(e, \varphi(N)) = 1$. Následně pomocí Euklidova algoritmu vypočítáme $d \in \mathbb{N},\ 1 < d < \varphi(N)$ (dešifrovací exponent) takové, že platí

$$ed \equiv 1 \pmod{\varphi(N)}.$$

  Nyní zveřejníme dvojici (e, N) jako veřejný klíč a ponecháme si $(d, p, q, \varphi(N))$ jako soukromý klíč.

- *Podpis zprávy m*
  Vypočítáme

$$s = m^d \pmod{N}$$

  a pošleme náš podpis s druhé straně.

- *Verifikace*
  Ověříme, zda platí

$$m = s^e \pmod{N}.$$

Tato rovnice by měla platit, protože

$$m = s^e \equiv (m^d)^e \equiv m^{de} \equiv m \pmod{N}.$$

Brzy po RSA byla vyvinuta další schémata digitálního podpisu. Nejznámější jsou Lamportův podpis, Merklův podpis, Rabinův podpis a nejdůležitější ElGamalův podpis.

ElGamalův podpis byl navržen v roce 1984 a jeho bezpečnost je založena na problému diskrétního logaritmu. Funguje následovně:

- *Generování klíče*
  Zvolíme si prvočíslo p a vypočítáme generátor g cyklické grupy $\mathbb{Z}_p^*$. Dále si zvolíme náhodné přirozené $x,\ 1 < x < p-1$, a vypočítáme

$$y = g^x \pmod{p}.$$

  Veřejný klíč je trojice (p, g, y) a soukromý klíč je hodnota x.

- *Podpis zprávy m*
  Zvolíme si náhodné k takové, že platí $0 < k < p-1$ a $NSD(k, p-1) = 1$. Dále si zvolíme si hašovací funkci a vypočítáme haš zprávy $h(m)$. Potom vypočítáme
  $$r \equiv g^k \pmod{p},$$
  $$s \equiv (h(m) - xr)k^{-1} \pmod{p-1}.$$
  A podpis je dvojice (r, s).

- *Verifikace*
  Podpis je platný, pokud
  $$g^{h(m)} \equiv y^r r^s \pmod{p}.$$

V posledních desetiletích byly navrženy různé varianty ElGamalova podpisu, především DSA a Schnorr, které jsou vysvětleny v této práci.

7. Implementace

Oba algoritmy jsem zkoušela naprogramovat, abych si ověřila časovou složitost. Také jsem naprogramovala jeden z útoků, kde používáme při podepisování dvou zpráv stejnou jednorázovou hodnotu.

7.1 Instalace

Program jsem implementovala v prostředí CodeBloks na operačním systému Windows 7. Při implementování jsem použila knihovny NTL, GMP a SHA1, které musíme vložit do složky „include“ a následně program „main“ zkompilujeme s přiloženým „sha1“.

7.2 Dokumentace

Přiložený program počítá digitální podpis pomocí DSA a Schnorrova algoritmu. Uživatel uvede v bitech, jak velké chce mít prvočíslo q, a pak vloží dvě zprávy bez mezer. Procedura „KEYGEN“ vygeneruje všechny klíče a pak je vypíše na obrazovku. Nejdříve si náhodně zvolí prvočíslo q o velikosti, která byla zadána, a následně si vyhledá takové prvočíslo p, aby platilo $q|p-1$. V procedurách „DSA“ a „SCHNORR“ se vypočte podpis a v následujících procedurách „overeniDSA“ a „overeniSCHNORR“ zkontroluje, zda byl podpis vygeneroval správně. V případě, že vše bylo v pořádku, program vypíše „podpis je spravne“ a v opačném případě vypíše „podpis je spatne“.

Ve skutečnosti se počítají dva podpisy DSA zadaných zpráv, ale se stejnou jednorázovou hodnotou k, a procedura „utok“ hledá tuto jednorázovou hodnotu k a následně i soukromý klíč x pomocí jednoho z útoků, který byl ve čtvrté kapitole vysvětlen. Tento útok je úspěšný pokaždé.

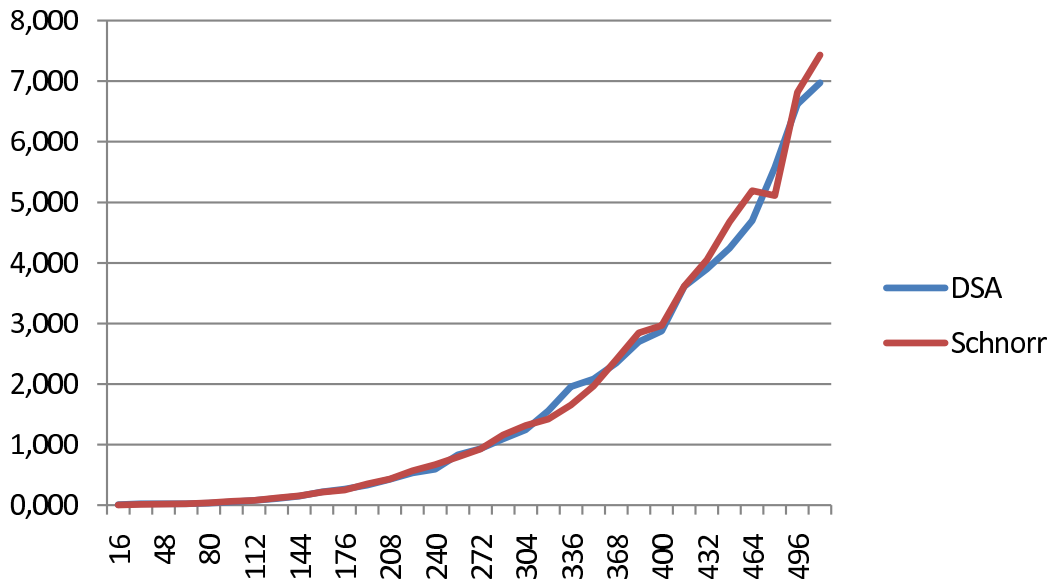
Poslední procedura je „mereni“. Sleduje, jak dlouho se počítají pro různé délky prvočísla q tyto podpisy pro oba algoritmy.

7.3 Výsledky

Níže v tabulce můžeme sledovat rychlost DSA a Schnorr. V grafu pod tabulkou je patrné, že opravdu časová složitost je $O(n^2)$. Ukázalo se, že rychlost obou algoritmů je téměř shodná, vetšinou však Schnorrův algoritmus pracuje rychleji.

velikost q v bitech	rychlost v sekundách		velikost q v bitech	rychlost v sekundách	
	DSA	Schnorr		DSA	Schnorr
16	0.004	0.001	272	0.93	0.924
32	0.023	0.009	288	1.091	1.157
48	0.02	0.015	304	1.34	1.314
64	0.023	0.022	320	1.555	1.422
80	0.034	0.035	336	1.954	1.654
96	0.053	0.06	352	2.08	1.964
112	0.075	0.077	368	2.343	2.401
128	0.109	0.117	384	2.696	2.844
144	0.148	0.151	400	2.876	2.967
160	0.219	0.214	416	3.61	3.617
176	0.263	0.249	432	3.899	4.049
192	0.329	0.349	448	4.242	4.677
208	0.424	0.428	464	4.696	5.19
224	0.53	0.568	480	5.576	5.113
240	0.591	0.669	496	6.62	6.818
256	0.829	0.795	512	6.972	7.432

Tabulka 7.1: časová složitost



Obrázek 7.1: srovnání časové složitosti

Seznam použité literatury

[1] David Stanovský (2010), *Základy algebry*. MatfyzPress, Učební text MFF UK.

[2] Bruce Schneier (1996), *Applied Cryptography: Protocols, Algorithms and Source Code in C. 2nd edition. [s.l.] : John Wiley ans Sons, Inc., c1996. 758 s. ISBN 0-471-11709-9.*

[3] Alfred J. Menezes, Paul C. van Oorschot a Scott A. Vanstone (1996), *Handbook of Applied Cryptography*. CRC Press Chapter 3, 9, 10 ,11. http://www.cacr.math.uwaterloo.ca/hac/

[4] *The SHA-3 Zoo*. http://ehash.iaik.tugraz.at/index.php/The_SHA-3_Zoo

[5] ODLYZKO, A. M , *Discrete Logarithms In Finite Fields And Their Cryptographic Significance*. ATT Bell Laboratories. http://www.dtc.umn.edu/ odlyzko/doc/arch/discrete.logs.pdf

[6] Vlastimil Klíma, *Vyřešena úloha diskrétního logaritmu ECC2-109*. http://crypto-world.info/news/index.php?prispevek=78&sekce=s

[7] Thorsten Kleinjung, *Discrete logarithms in GF(p)*. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0702&L=nmbrthry&T=0&P=194

[8] Joppe W. Bos and Marcelo E. Kaihara, *PlayStation 3 computing breaks* 2^{60} *barrier* 112*-bit prime ECDLP solved*. http://lacal.epfl.ch/page81774.html

[9] A.A. Puntambekar (2008), *Desing Analysis of Algorithms* .

[10] *FIPS PUB 186-3*. http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

[11] *Zákon c. 227/2000 Sb., o elektronickém podpisu*. http://www.mvcr.cz/clanek/zakon-c-227-2000-sb-o-elektronickem-podpisu.aspx

[12] Richard A. Mollin (2007), *An introduction to cryptography*. Str. 157-188.

[13] Martin Hlaváč (2010), *Cryptanalytic Attacks*. Učební text MFF UK. http://www.mhlavac.info/wp-content/uploads/downloads/2010/03/lec2.pdf

[14] Zhengjun Cao, Olivier Markowitch (2009), *Security Difference Between DSA and Schnorr's Signature*. Sborník z konference (International Conference on Networks Security, Wireless Communications and Trusted Computing. Print) Str. 201-204.

[15] Zhengjun Cao , *On the big gap between $|p|$ and $|q|$ in DSA* .