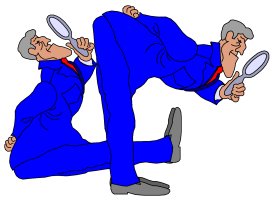
11. Autentičnost a digitální podpisy

Jan Paseka

Ústav matematiky a statistiky
Masarykova univerzita

14. prosince 2023

O čem to bude



Úvod I

FI V předchozích kapitolách jsme předvedli metody, které mohou pomoci proti **pasívnímu útoku**; pomocí zašifrování lze data pro nepovolané osoby udělat nečitelná. Téma této kapitoly je věnováno metodám proti **aktivnímu útoku**.

Kryptografický protokol specifikuje, jakým způsobem každá strana začíná a odpovídá na zprávy a to včetně chybných nebo ilegálních zpráv.

Protokol může rovněž specifikovat požadavky na nastavení jako je např. nastavení knihovny veřejných klíčů. Strana, která se řídí protokolem, bude ochráněna proti jistým specifikovaným nebezpečím i v tom případě, že ostatní strany se protokolem neřídí.

Úvod II

Je zřejné, že Mr. X může způsobit podstatně větší škodu, jestliže neumí pouze pasívně číst data, nýbrž je dokonce aktivně změnit.

Skutečně se u většiny dnešních aplikací v kryptologii požaduje autentičnost dat a ne jejich utajení.

Je zvykem rozlišovat 3 základní typy problémů, které vzniknou z různých variant aktivního útoku. Odpovídající "***bezpečnostní architektura***" byla vytvořena institucí ***International Standards Organisation (ISO)*** v "Security Addendum k referenčnímu modelu ISO".

Nejdříve je třeba ptát se, zda byla zpráva přenesena bez změny nebo zfalšování; jedná se o požadavek **integrity zprávy**.

Úvod III

Je-li Mr. X v pozici, že může měnit zprávy, nezbývá než doufat, že příjemce ***zpozoruje*** případnou změnu. Musí být schopen rozhodnout, zda byla zpráva změněna či nikoliv.

Druhý typ útoku je prvnímu podobný; zde je položen důraz na otázku, zda si příjemce může být jistý, že zpráva skutečně pochází od údajného odesilatele. Mluvíme pak o **autentičnosti zprávy**.

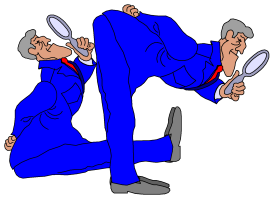
Poslední varianta je **autentičnost uživatele**: Může osoba dokázat svoji identitu? Příjemce potřebuje prostředek, aby se mohl přesvědčit o tom, že skutečně komunikuje s tou osobou, o které si myslí, že je s ní spojen.

Úvod II

Po pěti letech byla šifra AES schválena jako nejvhodnější z patnácti návrhů. Dne 26. května 2002 začala být ke svému účelu používána jako federální standard USA. AES je první šifra, dostupná široké veřejnosti, která je zároveň uznána Národní bezpečností agenturou NSA.

V současnoti se využívá k šifrování elektronické pošty, elektronického bankovnictví, různých druhů dálkové autentizace, čipových karet, elektronických peněz, přenosu hovorů v síti GSM, signálu wi-fi, bluetooth a satelitů.

O čem to bude



Symetrická autenticita I

Ochrana autentičnosti informace sestává z dvou následujících aspektů:

- ochrana původce informace neboli dle terminologie ISO autenticita původu dat,
- skutečnost, že informace nebyla změněna neboli dle terminologie ISO integrita informace.

První aspekt lze prezentovat tak, že je informace načítána např. z harddisku osobního počítače a my implicitně důvěřujeme zdroji informace.

Jiným aspektem je časování, umístění do fronty vzhledem k jiným zprávám a určení zprávy.

Až donedávna se obecně předpokládalo, že zašifrování informace je dostatečné k tomu, aby se prokázala její autenticita.

Symetrická autenticita II

Použitý argument byl ten, že pokud šifrový text dal po dešifrování smysluplnou informaci, tato by měla vzniknout od někoho, kdo zná tajný klíč, což garantuje autenticitu zprávy a odesílatele.

Ve dvou příkladech ukážeme, že tato víra není správná: ochrana integrity závisí na šifrovacím algoritmu a na módu, ve kterém je algoritmus použit.

Vernamova šifra, kde je náhodný klíč přičítán modulo 2 k šifrovému textu nám poskytuje perfektní bezpečnost, ale aktivní útočník může změnit libovolný bit zdrojového textu tím, že změní odpovídající bit šifrového textu.

Tato informace analogicky platí pro libovolnou přičítací proudovou šifru a pro OFB mód (Output FeedBack) každé blokové šifry.

Symetrická autenticita III

Částečně toto platí i pro případ, že šifra je použita CFB módu (Cipher FeedBack) nebo CBC módu (Cipher Block Chaining).

Je-li zdrojový text delší než jeden blok zašifrován pomocí blokové šifry v ECB módu, aktivní útočník může snadno přeuspořádat bloky.

Jiným příkladem zranitelnosti aktivním útočníkem je zdrojový text zašifrovaný pomocí CFB módu. Vzhledem k synchronizačním vlastnostem každá modifikace šifrového textu způsobí odpovídající modifikaci zdrojového textu a následně zkomolí následující části zdrojového textu. Poté co chyba opustí FB registr, bude šifrový text opět správně dešifrován.

Je-li ale modifikována poslední část šifrového textu, je zcela nemožné najít tuto modifikaci. Pokud se zkomolení vyskytne uprostřed zdrojového textu, lze chybu detekovat pomocí redundance.

Symetrická autenticita IV

V jiných módech (jako např. CBC mód) je každý šifrový text složitou funkcí předchozích bitů zdrojového textu a nějaké počáteční hodnoty.

Pokud modifikace jednoho bitu šifrového textu způsobí zkomolení t bitů zdrojového textu, pravděpodobnost, že že nový zdrojový text bude akceptován jako smysluplný, je rovna 2^{-tD}, kde D je redundance informace.

V případě přirozeného jazyka zakódovaného pomocí 5 bitů na charakter je redundance na bit $D \simeq 0.74$ a tato pravděpodobnost je rovna $2^{-22.2}$ pro $t = 30$.

Avšak, je-li $D = 0$ a zašifrování neposkytuje žádnou autenticitu, jsou všechny zprávy smysluplné a to nezávisle na šifrovacím algoritmu nebo na módu šifrování.

Symetrická autenticita V

To pak znamená, že útočník může modifikovat zprávy nebo padělat zprávy dle svého výběru.

Omezení je pak to, že útočník neví dopředu, co bude obsahem odpovídajícího zdrojového textu, ale pro mnohé aplikace lze takovýto útok považovat za zdroj vážných problémů.

Poznamenejme, že i v případě existence redundance se požaduje kontrola lidským faktorem nebo vhodným počítačovým programem.

Abychom zajistili integritu zprávy, je nutno přidat speciální redundanci, a je-li informace spojena s původcem zprávy, musí být použit v tomto procesu tajný klíč (to předpokládá spojení osoby a jejího klíče) nebo zvláštního kanálu pro zajištění integrity.

Symetrická autenticita VI

Můžeme pak identifikovat dvě základní metody.

- První metoda je analogická metodě symetrické šifry, kde utajení velkého množství dat je založeno na utajení a autenticitě krátkého klíče. V tomto případě autenticita informace závisí na utajení a autenticitě klíče.

  Abychom dosáhli tohoto účelu, informace se zkomprimuje na kvantitu pevné délky, kterou nazýváme ***hešovacím kódem***. Poté se hešovací kód připojí k informaci. Funkce, která provede tuto operaci komprese, se nazývá ***hešovací funkce***.

  Základní myšlenkou zabezpečení integrity je ***přidat redundanci*** k informaci. Přítomnost redundance dovoluje příjemci provést rozlišení autentické informace a podvodné informace.

Symetrická autenticita VII

Abychom garantovali původ dat, je nutno v procesu použít tajný klíč. Tajný klíč může být obsažen v procesu komprese nebo může být použit, aby ochránil hešovací kód a/nebo informaci. V prvním případě mluvíme o MACu (***Message Authentication Code***), zatímco v druhém případě se hešovací kód nazývá MDC (***Manipulation Detection Code***).

- Druhá metoda sestává na zajištění autenticity (jak integrity a autenticity původu) informace o autenticitě MDC. Typickým příkladem této metody je uživatel počítače, který počítá MDC pro všechny své důležité soubory. Může si pak uložit soubor všech MDC na disketě, kterou si bezpečně uschová. Pokud tyto soubory zašle vzdálenému příteli, může jednoduše poslat soubory a sdělit příteli po telefonu jejich MDC. Autenticita telefonního kanálu je zajištěna hlasovou identifikací.

Symetrická autenticita VIII

Přidání redundance není jistě dostatečné. Speciální důraz musíme klást na obranu proti útokům na vysoké úrovni, jako je například opakování autentifikované zprávy.
Oba případy nefungují, pokud si odesílatel a příjemce navzájem nedůvěřují. V prvním případě sdílejí stejný tajný klíč. Pokud jedna ze stran tvrdí, že informace byla změněna druhou stranou, nemůže soudce rozhodnout, kdo má pravdu, i když obě strany vydají společný tajný klíč. Druhý přístup může pouze zajistit nepřevzetí, pokud obě strany věří autenticitě MDC: v praxi je to však obtížné realizovat, protože obě strany mají podobný přístup ke kanálu.

Asymetrická autenticita I

Jestliže chceme být ochráněni proti vnitřnímu napadnutí, potřebujeme elektronický ekvivalent podpisu. V tomto případě třetí strana bude schopna rozlišit dvě strany a to na základě skutečnosti, že způsobilosti obou stran jsou různé.

Pojem digitálního podpisu byl zaveden W. Diffiem a M. Hellmanem.

Požadavky na elektronický podpis jsou, že ***podpis závisí na podepisované informaci*** (protože není fyzicky spjat s dokumentem) a že ***podepsaný je jediná osoba, která je schopna vytvořit podpis*** (to znamená, že nikdo jiný nemůže zfalšovat podpis tj. podepsaný nemůže zapřít, že informaci podepsal právě on).

Asymetrická autenticita II

Digitální podpisové schéma sestává z následujících prvků:

- inicializační fáze (např. generování klíče a obecné nastavení),
- procesu podpisu, kdy je vytvořen podpis,
- procesu verifikace, kdy příjemce (nebo soudce) ověří, zda je podpis správný.

Digitální podpis v tomto smyslu lze vytvořit pomocí zařízení bezpečných proti falšování, konvenčních jednosměrných funkcí nebo technik veřejného klíče.

Poznamenejme dále, že bylo definováno několik zobecnění – např. s různými stupni bezpečnosti a více hráči ve hře.

Asymetrická autenticita III

Příklady takovýchto jsou následující: ***libovolné podpisy***, kde proces podpis a verifikace zahrnuje interakci s třetí stranou, ***skupinové podpisy***, kde podpisující a/nebo kontroloři jsou členy skupiny, ***slepé podpisy***, kde podpisující podepíše "slepou " nebo " maskovanou" zprávu a ***neviditelné*** nebo ***nepopiratelné*** zprávy, kde lze podpis verifikovat pouze ve spolupráci s podpisujícím.

Message-Authentication-Code I

Připomeňme si, že při integritě a autentičnosti zprávy jde o to, abychom vyvinuli metody, které příjemci umožní rozhodnout, zda zpráva došla neporušená a autentická. K tomu potřebuje příjemce něco, s čím může být zpráva ověřena: Potřebuje dodatečnou informaci od odesilatele.

Takový informační blok se nazývá **kryptografický zkušební součet, kryptografický otisk prvku** neboli **Message-Authentication-Code**, zkráceně **MAC**.

Protokol k vytvoření a verifikaci kryptografického zkušebního součtu je založen na použití tajného klíče k, který je znám jak odesilateli tak příjemci, a kryptografickém algoritmu A, který budeme v dalším diskutovat.

Message-Authentication-Code II

Odesílatel neposílá pouze holou zprávu M, nýbrž dodatečně příslušný MAC; ten se vypočte pomocí klíče k algoritmem A ze zprávy M následovně:

$$\text{MAC} = A_k(M).$$

Poznamenejme, že M je odesíláno nezašifrované, protože cílem odesílatele není utajit obsah zprávy, nýbrž zprávu zabezpečit. Pokud chceme navíc důvěrnost, musí být m a MAC zašifrovány.

Nyní přijde na řadu příjemce. Jeho zájem je zjistit, zda přijatá zpráva souhlasí se zprávou odeslanou a zda skutečně pochází od uvedeného odesilatele.

Message-Authentication-Code III

Aby to provedl, simuluje proceduru odesilatele: Použije algoritmus A s klíčem k na přijatou zprávu M' a prověří, zda výsledek souhlasí s obrženým MACem.

Je-li $A_k(M') \neq MAC'$, ví příjemce, že se "***něco***" stalo: proto neakceptuje zprávu jako autentickou a odmítne ji.

Je-li ale $A_k(M') = MAC'$, může si být dostatečně jistý, že zpráva nebyla změněna. Přirozeně tato jistota závisí ve velké míře na kvalitě algoritmu A a velikosti množiny možných klíčů. Představy o MAC–mechanismu jsou následující:

- Podvodu ze strany Mr. X bude zamezeno, protože nezná klíč k. Musel by totiž spočítat odpovídající MAC pro svou zprávu.

Úvod
Integrita a autenticita
One-way
Uživatel
Symetrie
Asymetrie
MAC

Message-Authentication-Code IV

- Příjemce může pouze ***rozpoznat***, či je zpráva neporušená a autentická; v záporném případě nemá žádnou možnost zrekonstruovat původní zprávu. To znamená, že v tomto případě je nutný nový přenos zprávy.
- MAC–mechanismus je metoda k dosažení integrity a autentičnosti. Už jsme viděli, že lze poznat integritu. Pokud proběhne verifikace kladně, je příjemce rovněž přesvědčen o autentičnosti zprávy, protože odesílatel je jedinou jinou instancí, která zná tajný klíč.

Message-Authentication-Code V

Jaké algoritmy A můžeme použít k výpočtu MACu?

Okamžitá odpověď je jednoduchá: použijme jednoduše šifrovací algoritmus, přičemž MAC je kryptogram, který odpovídá zprávě M.

Odhlédneme-li od toho, že takovýto slabý algoritmus není vhodné doporučit, má tento návrh tu nevýhodu, že přenášená data jsou dvojnásobně delší než "vlastní zpráva".

Přirozeně každý MAC prodlouží zprávu, ale chtěli bychom délku tohoto dodatečného bloku držet v nějakých rozumných mezích.

V praxi používáme k výpočtu MACu také šifrovací algoritmus, ale ne přímo, nýbrž v tzv. **Cipher-Block-Chaining módu**.

Message-Authentication-Code VI

Představme si šifrovací algoritmus f (v praxi se většinou používá algoritmus DES nebo AES), který zobrazuje bloky zprávy složených z n znaků pomocí nějakého klíče K na bloky kryptogramu, rovněž složených z n znaků (typická hodnota je $n = 64$).

Abychom mohli vypočítat MAC, rozdělíme zprávu M do bloků $M_1, M_2, \ldots, M_s$ délky n.

Pak aplikujeme f na blok M_1 a obdržíme první blok kryptogramu $C_1 = f_K(M_1)$.

Potom přičteme C_1 k M_2 a položíme $C_2 = f_K(C_1 \oplus M_2)$.

Tento postup opakujeme až skončíme výstupem $C_s = f_K(C_{s-1} \oplus M_s)$, který vybereme za MAC.

Message-Authentication-Code VII

Takto vypočtený MAC má následující přednosti:

- MAC má pevnou délku *n* nezávislou na délce zprávy.
- MAC závisí na všech blocích zprávy.

Protože všechny možné zprávy jsou zkomprimovány na MACy pevné délky, má mnoho zpráv tentýž MAC.

To nepředstavuje žádný problém pro příjemce, protože ten nemusí rekonstruovat původní zprávu z MACu.

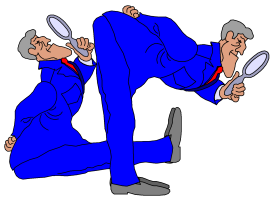
Ne všechny algoritmy jsou vhodné k tomu, aby byl Mr. X postaven před nepřekonatelné problémy.

Message-Authentication-Code VIII

Algoritmus pro výpočet MACu by měl mít následující vlastnosti.

1. Mělo by být prakticky nemožné najít pro daný MAC odpovídající zprávu (pokud tato vlastnost platí, nazýváme algoritmus **jednosměrnou** (one-way) **funkcí**). "Prakticky nemožné" znamená, že s dnešními metodami a počítači by vyřešení problému trvalo příliš dloho (několik století).
2. Mělo by být prakticky nemožné najít dvě zprávy, které mají tentýž MAC (jednosměrná funkce splňující tuto podmínku se nazývá **bezkolizní**).

O čem to bude



Definice jednosměrné funkce I

Je velmi obtížné podat precizní matematickou definici jednosměrné funkce. Neformálně je ***jednosměrná funkce*** funkce $f : S \to T$, kde S a T jsou množiny takové, že
(1) pro všechna $x \in S$ je $f(x)$ "***snadno***" vypočitatelné,
(2) máme-li k dispozici informaci, že $f(x) = y$, neexistuje žádný "***přiměřený***" způsob
jak získat (výpočtem) x pro "***dostatečně velké***" množství prvků y z T.

Pracovní slova zde jsou "***snadno***", "***přiměřeně***" a "***dostatečně velké***".

Je zřejmé, že je-li dáno $f(x)$, jeden způsob, jak získat x je prohledávat všechny možné hodnoty $x \in S$. Nepovažujeme to za přiměřené, protože S sestává obvykle z posloupnosti binárních řetězců délky $n \sim 200$.

Definice jednosměrné funkce II

Požadujeme, že výpočet pro nalezení x ze znalosti y je příliš dlouhotrvající nebo nákladný, kdykoliv y leží v "dosti velké" podmnožině množiny T.

Příklad. Elementárním příkladem kandidáta na jednosměrnou funkci je, pro dostatečně velké prvočíslo p, funkce $f(x)$, kde $f(x)$ je polynom nad tělesem $\mathbf{Z}_p$.

Pak je relativně snadné vypočíst $f(x)$ $(1 \leq x \leq p-1)$, ale obvykle je těžké nalézt řešení rovnice

$$f(x) = y.$$

Výše uvedená nepřesná definice znamená, že to, co je jednosměrná funkce, se mění s dobou.

Definice jednosměrné funkce III

Například, výpočet požadující milión instrukcí a 10 000 slov paměti nemohl být v roce 1950 považován za snadný, ale nyní by trval několik sekund na osobním počítači.

Tedy funkce považovaná v roce 1950 za jednosměrnou nemusí být za ni považovaná nyní.
Jedna metoda podání formální definice by mohlo být užitím fyzikálního přístupu.

Např. 10^{60}-bitová paměť vždy zůstane nedosažitelnou, protože i kdybychom potřebovali pouze jednu molekulu na bit paměti, její konstrukce by vyžadovala více hmoty než existuje v slunečním systému.

Podobně, termodynamika nám dává omezení maximálně 10^{70} operací, které lze provést s využitím celkové energie slunce.

Nižší úroveň nedosažitelnosti je použití myšlenek výpočetní složitosti.

Definice jednosměrné funkce IV

Nejdříve uvažujme některé z vlastností, které bychom rádi požadovali po jednosměrné funkci.

(I) Výpočet $f(x)$ z x musí být přiměřený: vyjádříme to tím, že f je vypočitatelná v polynomiálně omezené době (říkáme, že $f \in \mathrm{P}$).

(II) Výpočet f^{-1} nesmí být snadný; budeme tudíž požadovat, že není znám žádný algoritmus pro výpočet f^{-1} v polynomiálně omezené době.

(III) Třetí podmínka bude tzv. ***upřímnost*** funkce tj., že existuje polynom p splňující $|x| \leq p(|f(x)|)$.

Poslední podmínka je technická podmínka pro vyloučení funkcí jako

$$f(x) = \lceil \log \log x \rceil,$$

která zcela jistě splňuje (I) a (II), ale kterou bychom nemohli obvykle považovat za jednosměrnou funkci.

Definice jednosměrné funkce V

Funkce f splňující (I),(II) a (III) se nazývá slabě jednosměrná funkce.

Příklad. Nechť $\mathbf{I}_k$ značí množinu všech k-bitových přirozených čísel tj.

$$\mathbf{I}_k = \{2^{k-1}, \ldots, 2^k - 1\} \qquad (k = 1, 2, \ldots).$$

Nechť $\mathbf{S}_k = \mathbf{I}_k \times \mathbf{I}_k$ a nechť $f : \mathbf{S}_k \to \mathbf{Z}^+$ je definována jako

$$f(m, n) = m \cdot n.$$

Položíme-li $\mathbf{S} = \bigcup\{\mathbf{S}_k : 1 \leq k < \infty\}$ a rozšíříme-li f na $\mathbf{S}$, získáme slabě jednosměrnou funkci. Přitom v současné době není známo, že by inverzní funkce ležela v $\mathbf{P}$.

Definice jednosměrné funkce VI

Není lehké najít matematickou explicitní definici jednosměrné funkce. Uveďme následující příklad.

Příklad. Buď F šifrovací algoritmus, který zobrazuje zprávu M pomocí klíče K na kryptogram C, $F_K(M) = e(M, K) = C$ a předpokládejme, že $\mathbf{M} \subseteq \mathbf{K}$.

Změňme trochu tuto funkci. Zafixujme za tímto účelem (ne tajnou) ***zprávu*** M_0 (např. $M_0 = 00\ldots0$); tato "zpráva" se objeví v algoritmu místo obvyklé zprávy, nehraje ale roli variabilní zprávy.

Variabilní zpráva se vloží do algoritmu F na místě klíče. Krátce: uvažujeme funkci

$$f = e(M_0, -) : \mathbf{M} \to \mathbf{C}.$$

Tvrdíme pak, že f je jednosměrná funkce.

Definice jednosměrné funkce VII

Představme si, že Mr. X zná jak M_0 tak i C a chtěl by najít M. V řeči šifrovacího algoritmu F to lze vyjádřit následovně: Mr. X zná sobě odpovídající dvojici zpráva-kryptogram (M_0, C) a chtěl by vypočíst klíč.

Je-li algoritmus F kryptologicky bezpečný, je rezistentní proti tomuto known-plaintext útoku a proto je f jednosměrná funkce.

Položme si otázku, zda lze ***matematicky dokázat***, že tato funkce f je jednosměrná. Odpověď zní: ***Ne!***

Matematici nemohli ještě o žádné funkci dokázat, že je jednosměrná.

Definice jednosměrné funkce VIII

To znamená, že neznáme žádnou funkci, jejíž funkční hodnoty lze spočítat v polynomiálně omezené době, ale která při výpočtu funkce inverzní potřebuje exponenciální dobu.

Nevíme tedy, zda teoreticky jednosměrné funkce existují. Pro praktické účely mají ale výše popsané funkce dostatečně dobré vlastnosti.

Totiž, kdybychom to uměli, ***uměli bychom dokázat, že*** $P \neq NP$.

Zanedbatelné funkce I

Připomeňme, co se snažíme zachytit v následujících definicích.

- Za prvé, když mluvíme o efektivním šifrovacím nebo dešifrovacím algoritmu, obvykle máme na mysli takový, který běží velmi rychle a šifruje data rychlostí řekněme 10–100 počítačových cyklů na byte dat.
- Za druhé, když mluvíme o efektivním protivníkovi, obvykle máme na mysli algoritmus, který běží během nějaké velké, ale stále proveditelné doby (a dalších zdrojů). Obvykle se předpokládá že protivník, který se snaží prolomit kryptosystém, je ochoten utratit mnohem více zdroje než uživatel kryptosystému. Tedy 10 000 paralelně běžících počítačů 10 let může být považováno za horní hranici toho, co je proveditelně vyčíslitelné trpělivý a finančně dobře situovaným protivníkem.

Zanedbatelné funkce III

- Za třetí, když mluvíme o výhodě protivníka jako o ***zanedbatelné***, myslíme tím, že ji lze považovat za tak malou, že ji lze pro všechny praktické účely chápat rovnou nule.

Přijmeme rovnítko mezi pojmem efektivního algoritmu a pojmem (pravděpodobnostní) polynomiální-časový algoritmus.

V dobrém i ve zlém nám to dává formální rámec, který je nezávislý na konkrétních detailech jakéhokoli konkrétního modelu výpočtu.

Zanedbatelné funkce IV

V rámci celé kapitoly budeme automaticky předpokládat, že všechny polynomy jsou ***pozitivní***, tj. $p(k) \geq 1$ pro všechna přirozená čísla $k \geq 1$.

Definition 3.1

Funkce $r : \mathbf{N} \to \mathbf{R}_{>0}$ je ***zanedbatelná***, jestliže pro každý (pozitivní) polynom $p : \mathbf{N} \to \mathbf{N}$, existuje celé číslo k_0 takové, že $r(k) < 1/p(k)$ pro $k \geq k_0$.
Budeme používat označení $\mathrm{neg}(\cdot)$, které označuje libovolnou zanedbatelnou funkci.

Takže zanedbatelná funkce bude časem menší, než je inverze jakéhokoliv (pozitivního) polynomu.

Zanedbatelné funkce V

Zaveďme si pro funkce $r, s : \mathbf{N} \to \mathbf{R}_{>0}$ následující relaci:

$r \leq_{\text{ev}} s$ tehdy a jen tehdy, když

$\exists\, k_0 \in \mathbf{N}$ takové, že $r(k) \leq s(k)$ pro $k \geq k_0$.

Tato relace je evidentně reflexivní a tranzitivní a funkce $r : \mathbf{N} \to \mathbf{R}_{>0}$ je zanedbatelná právě tehdy, když $r \leq_{\text{ev}} (\cdot)^{-c}$ pro každé přirozené číslo c; zde $(\cdot)^{-c}$ je funkce $k \mapsto k^{-c}$.

To je ale evidentně ekvivalentní podmínce, že když $r \leq_{\text{ev}} (\cdot)^{-c}$ pro každé kladné reálné číslo c.

Zanedbatelné funkce VI

Alternativní charakterizace zanedbatelné funkce, se kterou se snad lépe pracuje, je následující:

Věta 3.2

Funkce $r : \mathbf{N} \to \mathbf{R}_{>0}$ je zanedbatelná právě tehdy, když pro všechna $c > 0$ máme

$$\lim_{n \in \mathbf{N}} r(n) \cdot n^c = 0.$$

Důkaz.

Buď $r : \mathbf{N} \to \mathbf{R}_{>0}$ zanedbatelná funkce, $c > 0$. Pak

$$0 \leq \lim_{n \in \mathbf{N}} r(n) \cdot n^c \leq \lim_{n \in \mathbf{N}} n^{-(c+1)} \cdot n^c = \lim_{n \in \mathbf{N}} n^{-1} = 0.$$

Obráceně z definice limity existuje k_0 tak, že pro $n \geq k_0$

$$r(n) \leq \frac{1}{2n^c} < n^{-c}.$$

Zanedbatelné funkce VII

Příklad 3.3

1. *Příkladem zanedbatelných funkcí jsou* $2^{-n}, 2^{-\sqrt{n}}, n^{logn}$.
2. *Příkladem funkcí, které nejsou zanedbatelné, jsou* $\frac{1}{1000n^5+n^2 logn}$, $\frac{1}{n^{1000}}$.

Zanedbatelné funkce VIII

Tvrzení 3.4

Funkce r je zanedbatelná tehdy a jen tehdy, pokud existuje zanedbatelná funkce s taková, že $r \leq_{ev} s$.

Důkaz.

Jestliže r je zanedbatelná, pak stačí položit $s = r$.

Naopak předpokládejme, že existuje zanedbatelná funkce s taková, že $r \leq_{ev} s$. Chceme ukázat, že r je zanedbatelná.

Nechť $c \in \mathbf{N}$. Pak $r \leq_{ev} s$ a $s \leq_{ev} (\cdot)^{-c}$.
Z tranzitivity pak i $r \leq_{ev} (\cdot)^{-c}$. Tedy r je zanedbatelná. ∎

V teoretické kryptografii se tímto formalizuje pojem pravděpodobnosti úspěchu protivníka být "příliš malý, aniž by nám to vadilo" tím, že požadujeme, aby ***tato pravděpodobnost byla zanedbatelnou funkcí parametru zabezpečení***.

Zanedbatelné funkce IX

Uvědomme si, že funkce $r : \mathbf{N} \to \mathbf{R}$ ***není zanedbatelná***, právě když existuje pozitivní polynom $p(n)$ a nekonečně mnoho hodnot $k \in \mathbf{N}$ tak, že $r(k) \geq 1/p(k)$.
Následující výsledek nám říká, že naše definice zanedbatelnosti perfektně zapadá do myšlenky, že pouze polynomiální časové výpočty jsou proveditelné.
Říká tedy, že pokud algoritmus má zanedbatelnou naději na úspěch, pak jeho opakování polynomiálně mnohokrát nemůže změnit tuto skutečnost.

Tvrzení 3.5

Pokud je pravděpodobnost toho, že algoritmus E bude úspěšný (v dané výpočetní úloze) na vstupech velikosti k, **zanedbatelná** *(v k), pak pravděpodobnost toho, že uspěje alespoň jednou, když se mnohokrát polynomiálně opakuje,* **je také zanedbatelná**.

Zanedbatelné funkce X - Důkaz Věty 3.5

Důkaz.

Předpokládejme, že existuje pozitivní polynom $q(n)$ tak, že pravděpodobnost alespoň jednoho úspěchu na vstupech velikosti k, pokud je algoritmus E opakován $q(k)$-krát, není zanedbatelná.

Stačí pak ukázat, že pravděpodobnost úspěchu $r(k)$ algoritmu E na vstupech velikosti k nemohla být zanedbatelná.

Ale to znamená, že existuje pozitivní polynom $p(n)$ a nekonečně mnoho hodnot $k \in \mathbf{N}$ tak, že $q(k)r(k) \geq 1/p(k)$.

To je ale ekvivalentní s tím, že pro nekonečně mnoho hodnot $k \in \mathbf{N}$ platí $r(k) \geq 1/(q(k)p(k))$.

Tedy r není zanedbatelná, spor. ∎

Zanedbatelné funkce XI

Okamžitým důsledkem je pak následující tvrzení

Důsledek 3.6

Buď $negl_1(n)$ a $negl_2(n)$ zanedbatelné funkce, $p(n)$ pozitivní polynom. Pak

1. *$p(n)\,negl_1(n)$ je zanedbatelná funkce.*
2. *Součet $negl_1(n) + negl_2(n)$ je zanedbatelná funkce.*

Důkaz.

První část okamžitě plyne z důkazu Věty 3.5. Druhá část plyne z toho, že pro $c \in \mathbf{N}$ existuje $k_0 \in \mathbf{N}$ takové, že

$$\tfrac{1}{2}\left(negl_1(k) + negl_2(k)\right) \leq k^{-c}$$

pro $k \geq k_0$. Ale funkce $2\tfrac{1}{2}\left(negl_1(n) + negl_2(n)\right) = negl_1(n) + negl_2(n)$ je dle první části zanedbatelná. ∎

Zanedbatelné funkce XII

Než budeme pokračovat, pokusme si vysvětlit, co přesně znamená "inverze" $f(x)$?

Vzhledem k tomu, že někdy budeme muset vzít v úvahu funkce, které nejsou prosté, bude to znamenat, že pro y, které je tvaru $y = f(x)$, najdeme nějaké z, které bude splňovat $f(z) = y$. Označme pak

$$f^{-1}(f(x)) = \{z \in \{0,1\}^* \mid f(z) = f(x)\}.$$

Některé funkce jsou ale těžko invertovatelné ze zcela triviálního důvodu: délka každého takového z je mnohem delší, než je délka $f(x)$.

Jednosměrnou funkci by ale mělo být těžké invertovat, protože je těžké najít takovéto z, ne proto, že pokud je takovéto z nalezeno, trvá příliš dlouho jeho vlastní zápis.

Zanedbatelné funkce XIII

Příklad 3.7

Uvažme například funkci $f : \{0,1\}^* \to \{0,1\}^*$, $f(x) =$ *posledních* $\log_2 |x|$ *bitů binárního zápisu čísla* x.

Je zřejmé, že jakékoliv z *je exponenciálně delší než* $f(z)$ *samo o sobě. Tedy žádný algoritmus nemůže invertovat* $f(x)$ *v polynomiálním čase.*

Chceme-li se vyhnout tomuto problému, budeme předpokládat, že vstup do jakéhokoli invertující algoritmu pro $f(x)$ bude zahrnovat délku x, zakódovanou do posloupnosti jedniček.

Takže pokud $|x| = k$ potom se jako vstup invertujícího algoritmu bere dvojice $(f(x), 1^k)$ a výstupem by měl být prvek $z \in f^{-1}(f(x))$.

To nám zaručuje, že alespoň jedno takovéto z může být zapsáno v polynomiálním čase.

Jednosměrné funkce I

Pak lze přeformulovat definici jednosměrné funkce $f : \{0,1\}^* \to \{0,1\}^*$ následovně:

(I) Výpočet $f(x)$ z x musí být ***přiměřený***: vyjádříme to tím, že f ***je vypočitatelná v polynomiálně omezené době***.

(II) Pro každý pravděpodobnostní algoritmus A pracující v polynomiálním čase, je ***pravděpodobnost*** toho, že A úspěšně provede inverzi $f(x)$, pro náhodně vybrané $x \in \{0,1\}^k$, ***zanedbatelná***. (II) lze pak přepsat následovně:

(II') Pro každý pozitivní polynom $q(n)$ a každý pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase, platí

$$P(\{A(f(x), 1^k) \in f^{-1}(f(x)) \mid x \in \{0,1\}^k\}) \leq \frac{1}{q(k)}$$

pro dostatečně velké k.

Jednosměrné funkce II

Poznamenejme, že výše uvedený přepis platí za předpokladu, že uvažujeme pouze spočetnou množinu takovýchto algoritmů. Přitom (II') lze chápat jako lze chápat jako ***uniformní jednosměrnost funkce*** f.

Buď v dalším f vypočitatelná v polynomiálně omezené době. Označme pak pro každý pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase a pro každé $k \in \mathbb{N}$ (což chápeme jako bezpečnostní parametr) hodnotu

$$\mathbf{Inv}_A(k) = P(\{A(f(x), 1^k) \in f^{-1}(f(x)) \mid x \in \{0,1\}^k\}).$$

Máme tedy pro každý pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase přiřazenou funkci $\mathbf{Inv}_A : \mathbb{N} \to \mathbb{R}_{>0}$.

Jednosměrné funkce III

Z výše uvedeného plyne, že:

f ***je jednosměrná*** právě tehdy, když pro každý pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase je funkce $\mathbf{Inv}_A$ ***zanedbatelná***.

Tedy funkce ***není jednosměrná***, pokud existuje pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase, který má ***nezanedbatelnou pravděpodobnost úspěchu***.

Řekneme, že f je ***uniformně jednosměrná*** právě tehdy, když existuje zanedbatelná funkce δ tak, že $\mathbf{Inv}_A \leq_{\text{ev}} \delta$ pro každý pravděpodobnostní algoritmus A na provedení inverze pracující v polynomiálním čase.

Jednosměrné funkce IV

Za rozumných podmínek lze dokázat, že f je ***uniformně jednosměrná*** právě tehdy, když f je ***jednosměrná***.

Přitom celý důkaz je veden pomocí zanedbatelných funkcí. Navíc lze zavést následující pojmy, které mají využití i v jiných partiích kryptografie.

Nechť $\mathbf{F} = \{F_i \mid i \in \mathbb{N}\}$ je soubor funkcí z $\mathbb{N}$ do $\mathbb{R}$.

Budeme uvažovat dvě definice pojmů "***zanedbatelnosti***" pro soubor **F**.

První definice je jednoduchá: stačí požadovat, aby jednotlivé funkce, pokud jsou posuzovány jednotlivě, byly zanedbatelné.

Formálně řečeno, **F** je ***bodově zanedbatelný***, je-li F_i zanedbatelná pro každé i z $\mathbb{N}$.

Jednosměrné funkce V

Druhou možností je požadovat, aby byl soubor **F** "jednotně" zanedbatelný tím způsobem, že funkce F_i odpovídají nějakému společnému meznímu bodu.

Formálně řečeno, **F** je ***uniformně zanedbatelný***, pokud existuje zanedbatelná funkce δ (nazývaná též ***mezní bod***) tak, že $F_i \leq_{\text{ev}} \delta$ pro všechna i z $\mathbb{N}$.

To znamená, že, pro všechna i z $\mathbb{N}$,

$$F_i(n) \leq \delta(n)$$

od dostatečně velkého n_0.

Lze dokázat, že oba pojmy zanedbatelnosti jsou ekvivalentní. Navíc nalezená zanedbatelná funkce δ je neklesající.

Aplikace na kryptografické pojmy I

Jak to nyní souvisí s jednocestnými funkcemi?

Nechť $\mathcal{I} = \{A_i \mid i \in \mathbb{N}\}$ je výčet všech pravděpodobnostních algoritmů A na provedení inverze funkce f pracující v polynomiálním čase. Vzhledem k tomu je $\mathcal{I}$ spočetná.

Pro každé $i \in \mathbb{N}$ definujeme funkci F_i tak, že $F_i(n) = \mathbf{Inv}_{A_i}(n)$.

Nechť $\mathbf{F} = \{F_i \mid i \in \mathbb{N}\}$.
Pak vidíme, že f je ***jednosměrná*** tehdy a jen tehdy, pokud soubor **F** je ***bodově zanedbatelný***.

Z ekvivalence obou pojmů máme, že f je ***jednosměrná*** tehdy a jen tehdy, pokud soubor **F** je ***uniformně zanedbatelný***.

Aplikace na kryptografické pojmy II

Teď, když to vidíme, je jasné, že totéž platí pro skoro jakýkoliv šifrovací primitiv.

Asymptotická definice bezpečnosti pro všechny primitivy má následující podobu.

Pro každého "nepřítele" A a jakoukoli hodnotu parametru zabezpečení n bude přiřazena pravděpodobnost úspěchu $\mathbf{Succ}_A(n)$, a to na základě nějakého experimentu. (Pro tuto chvíli bude protivník algoritmus).

Primitiv se bude nazývat "***bezpečný***", pokud pro každého útočníka A je funkce $\mathbf{Succ}_A$ ***zanedbatelná***.

Aplikace na kryptografické pojmy III

Nechť nyní bude $\mathbf{A} = \{A_i \mid i \in \mathbb{N}\}$ výčet všech možných protivníků. (Vzhledem k tomu je počet protivníků spočetný.)

Nechť $F_i(n) = \mathbf{Succ}_{A_i}(n)$.

Pak vidíme, že soubor $\mathbf{F} = \{F_i \mid i \in \mathbb{N}\}$ je bodově zanedbatelný.

Z předchozího dostáváme, že existuje ***zanedbatelná funkce*** δ tak, že pravděpodobnost úspěchu jakéhokoliv protivníka je od jisté výše parametru zabezpečení n nižší než δ.

Tedy k našemu primitivu je přiřazena specifická úroveň zabezpečení, které se všichni protivníci musí přizpůsobit.

Aplikace na kryptografické pojmy IV - Příklad

Nechť p je prvočíslo, g je primitivní kořen mod p a $x \in \mathbb{Z}_p^*$. Definujme

$$\text{dexp}(p, g, x) = (p, g, g^x \bmod p).$$

Funkci $\text{dexp}(p, g, x)$ ***lze snadno spočítat***, protože umocňování mod p může být provedeno v polynomiálním čase. Ale ***jak obtížné je ji invertovat***?

Definujeme "***inverzní***" funkci dexp jakožto

$$\text{dlog}(p, g, y) = x,$$

kde $y = g^x \bmod p$.

Aplikace na kryptografické pojmy V - Příklad

Všimněte si, že inverzní funkce dexp by opravdu měla dát na výstupu trojici (p, g, x), je však zjevně snadné najít p a g dané trojicí (p, g, y), zásadní 'problém' v invertování dexp spočívá v problému nalezení x.

Výpočet funkce dlog je známý jako ***problém diskrétního logaritmus***. Věří se, že je nesmírně těžký.

V současné době nejefektivnější algoritmus pro tento problém je založen na algoritmu ***Number Field Sieve*** pro faktorizaci, a za přijatelných předpokladů se očekává doba $O(exp(c(ln\ p)^{1/3}(ln\ ln\ p)^{2/3}))$.

I když si myslíme, že problém diskrétního logaritmu je obtížný, nevíme, zda je to pravda.

Aplikace na kryptografické pojmy VI - Příklad

Pokud chceme založit kryptografické protokoly na 'obtížnosti' problému diskrétního logaritmu, potřebujeme formulovat přesně předpoklad nepoddajnosti, exaktně popisující, jak věříme (nebo doufáme), že je problém diskrétního logaritmus obtížný.

To říká, že každý odpovídající protivník (pravděpodobnostní algoritmus pracující v polynomiálním čase) má pouze zanedbatelnou šanci vyřešení problému diskrétního logaritmu pro náhodný výběr.

Předpoklad diskrétního logaritmu

Pro jakýkoli kladný polynom $q()$ a jakýkoliv pravděpodobnostní polynomiální časový algoritmus A platí pro k dostatečně velké:

$$P[A(p, g, y) = \text{dlog}(p, g, y)] < \frac{1}{q(k)},$$

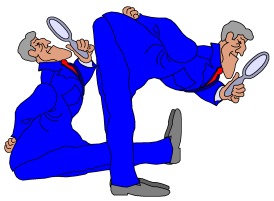
kde p je náhodné k-bitové prvočíslo, g je náhodný primitivní kořen mod p a x je náhodný prvek z $\mathbf{Z}_p^*$.

Aplikace na kryptografické pojmy VII

Tvrzení 3.8

Za Předpokladu diskrétního logaritmu je dexp silná jednosměrná funkce.

O čem to bude



Typy identifikace I

Základní úloha bezpečnostní techniky je spolehlivá identifikace osob tj. autentizace.

Dříve to byl výlučně proces mezi dvěma lidmi, který je dnes rozšířen na proces mezi člověkem a počítačem.

Přitom vznikají přirozeně problémy; ale, jak uvidíme, lze i s počítačem provádět velmi dobrou autenticitu uživatele.

Nejprve si připomeňme, jak je tento problém řešen mezi lidmi. Zásadně lze lidi poznávat podle následujících znaků:

- Osobu lze identifikovat podle ***charakteristických vlastností.***
- Osobu lze identifikovat podle ***vlastnictví***.
- Osobu lze identifikovat podle ***znalosti***.

Typy identifikace II

První mechanismus je v běžném životě používán tak často, že si ho ani nejsme vědomi: poznáváme naše známé podle jejich obličeje, hlasu, chůze atd. V jistých situacích se používají za účelem obzvlášť spolehlivé identifikace otisky prstů.

Následující metody se používají jen ve speciálních situacích. V druhém kole kupónové privatizace bylo nutno předložit občanský průkaz, kde je uvedeno české občanství; při přechodu hranic se musíme prokázat pasem; platíme-li kreditní kartou, lze ověřit identitu jejím vlastnictvím atd.

Identifikace na základě znalosti je prováděna velmi zřídka – ačkoliv tento mechanismus je znám již z nejstarších dob (vojáci musí znát současné heslo, aby se mohli identifikovat, když si chce policie ověřit, zda je unesená osoba ještě živá, položí otázky, které může zodpovědět pouze tato osoba).

Typy identifikace III

Při procesu autenticity mezi člověkem a počítačem je situace zcela jiná.

Autenticita na základě znalosti je mechanismus, který lze nejjednodušeji realizovat; autenticita na základě vlastnictví je rovněž možná.

Naproti tomu je autenticita podle charakteristických vlastností velmi komplexní a používá se pouze při aplikacích, které vyžadují extrémně vysokou bezpečnost.

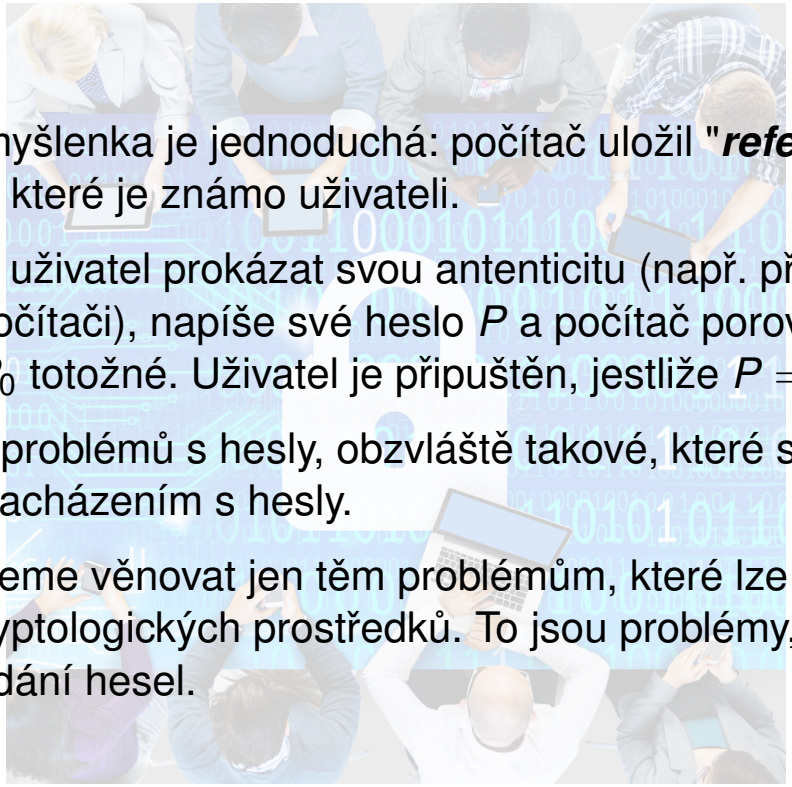
Proto se budeme výhradně zabývat s metodami založenými na znalosti resp. vlastnictví. Při použití těchto metod má uživatel nějaké tajemství a počítač se chce přesvědčit, že osoba má skutečně jí odpovídající tajemství.

Typy identifikace IV

Klasická metoda, se kterou se osoba prokáže stroji, je založena na heslech. ***Heslo*** je libovolná posloupnost znaků (písmena, číslice, speciální znaky), která je výlučným tajemstvím osoby a počítače. To znamená, že ***v ideálním případě*** znají jen uživatel a počítač příslušné heslo.

Identifier/Name	**IS MU - Login**	
Importance	5/5	
Difficulty	1/5	
Actor(s)	Generalized User	
Goal	To allow the user to access the system.	
Preconditions	The user is at the login page.	
Summary	Will validate the users name and password and subsequently give them access to the system.	
Steps	1. User provides username and password.	2. System directs user to main system page.
Postcondition	**Success:** User is logged in. **Failure:** The system remains at the login state.	

Typy identifikace V



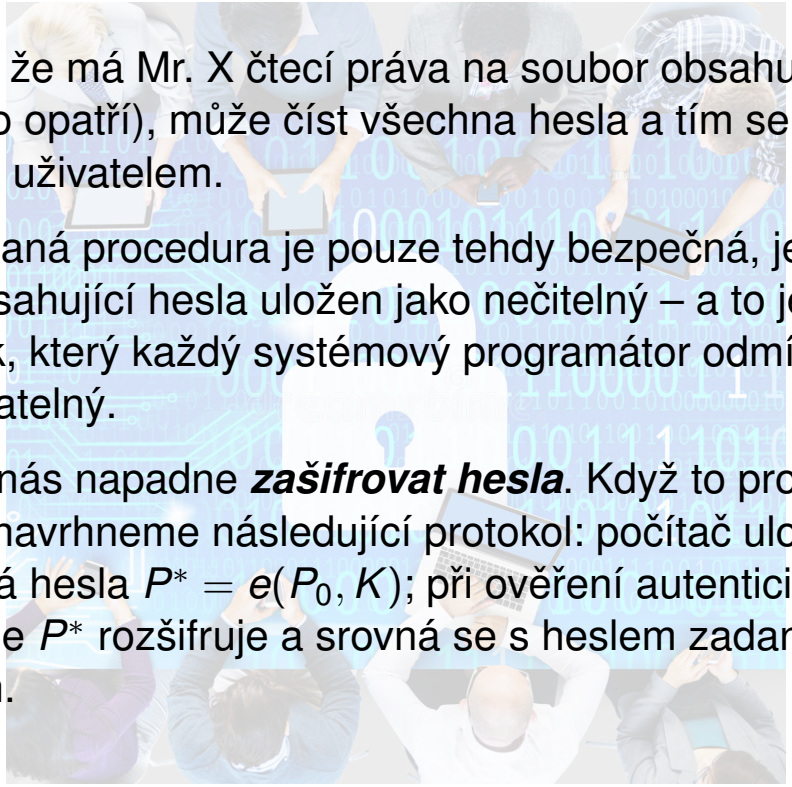
Základní myšlenka je jednoduchá: počítač uložil "***referenční heslo***" P_0, které je známo uživateli.

Když chce uživatel prokázat svou antenticitu (např. přístup k danému počítači), napíše své heslo P a počítač porovná, zda jsou P a P_0 totožné. Uživatel je připuštěn, jestliže $P = P_0$.

Je mnoho problémů s hesly, obzvláště takové, které souvisí se špatném zacházením s hesly.

My se budeme věnovat jen těm problémům, které lze řešit pomocí kryptologických prostředků. To jsou problémy, které se týkají ukládání hesel.

Typy identifikace VI



V případě, že má Mr. X čtecí práva na soubor obsahující hesla (nebo si ho opatří), může číst všechna hesla a tím se stát libovolným uživatelem.

Výše popsaná procedura je pouze tehdy bezpečná, jestliže je soubor obsahující hesla uložen jako nečitelný – a to je požadavek, který každý systémový programátor odmítne jako nerealizovatelný.

Přirozeně nás napadne ***zašifrovat hesla***. Když to provedeme "***naivně***", navrhneme následující protokol: počítač uloží zašifrovaná hesla $P^* = e(P_0, K)$; při ověření autenticity uživatele se P^* rozšifruje a srovná se s heslem zadaným uživatelem.

Typy identifikace VII

Tato procedura má jasnou výhodu. Dokonce i když Mr. X může číst soubor obsahující hesla, může je číst pouze zašifrovaná. Protože ale není schopen je dešifrovat, nemůže simulovat jednotlivého uživatele vložením správného hesla.

Je tento argument správný? Přirozeně jsme podstatně redukovali množinu tajných dat; namísto neohraničeného počtu hesel, která mají být tajně uložena, musíme v šifrovacím modelu jen uložit tajně klíč K.

To je nepochybně velká výhoda, ale není to žádné principiální řešení problému: Pokud si $\mathrm{Mr.\ X}$ opatří klíč K, může dešifrovat všechna zašifrovaná hesla a má tak stejně jako předtím možnost stát se libovolným uživatelem.

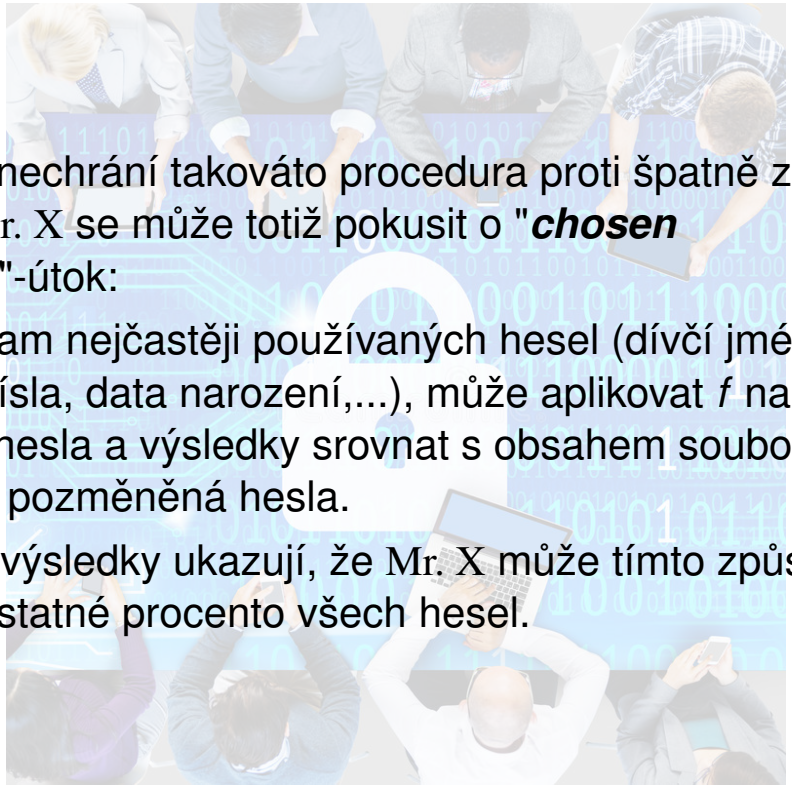
Typy identifikace VIII

Po takovéto analýze není pro tvůrce přístupového systému založeného na heslech těžké vyjádřit své přání: chtěl by mít k dispozici funkci, která "***zašifruje***" hesla bez toho, že by použila tajný klíč – to není nic jiného, než naše známá jednosměrná funkce.

Nyní použijme takovouto jednosměrnou funkci f. V paměti počítače uložme hodnotu $P^{\#} = f(P_0)$. Během procedury ověření autenticity se aplikuje f na posloupnost znaků P, kterou vloží uživatel. Pak počítač prověří, zda platí $P^{\#} = f(P)$.

Přednost této metody je zřejmá: soubor obsahující hesla je nedešifrovatelný a není žádné tajmeství. Je pozoruhodné, že jednosměrné funkce byly vyvinuty proto, aby ukládaly hesla v nečitelné formě; tato metoda byla v šedesátých letech navržena a realizována Rogerem **Needhamem** z University Cambridge.

Typy identifikace IX

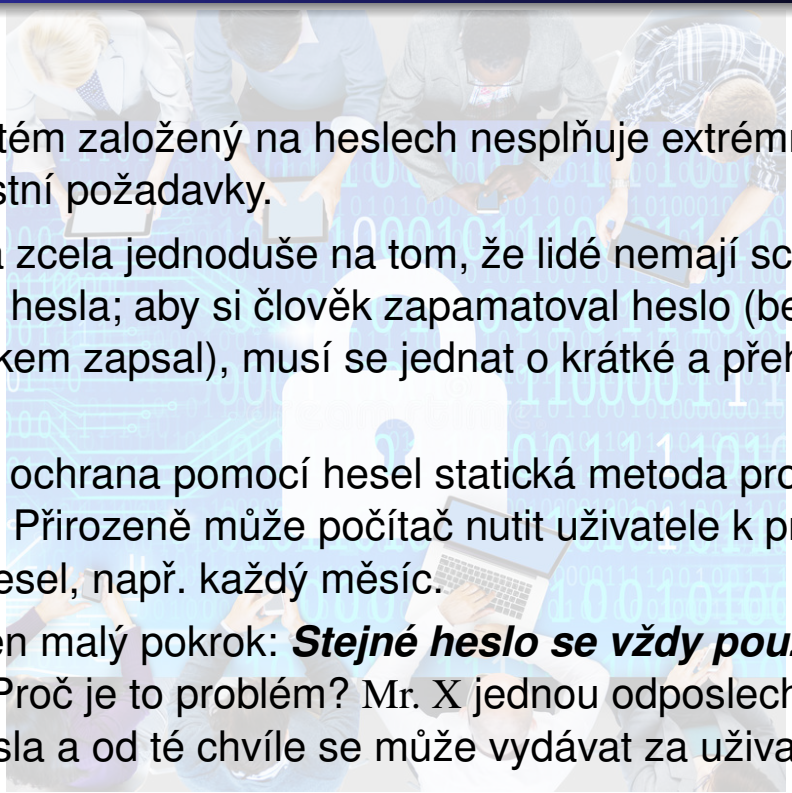


Přirozeně nechrání takováto procedura proti špatně zvoleným heslům. Mr. X se může totiž pokusit o "***chosen password***"-útok:

Má-li seznam nejčastěji používaných hesel (dívčí jména, telefonní čísla, data narození,...), může aplikovat f na tato populární hesla a výsledky srovnat s obsahem souboru obsahující pozměněná hesla.

Empirické výsledky ukazují, že Mr. X může tímto způsobem získat podstatné procento všech hesel.

Typy identifikace X - Čipové karty



Žádný systém založený na heslech nesplňuje extrémně vysoké bezpečnostní požadavky.

To spočívá zcela jednoduše na tom, že lidé nemají schopnost si pamatovat hesla; aby si člověk zapamatoval heslo (bez toho, že by si je bokem zapsal), musí se jednat o krátké a přehledné slovo.

Mimo to je ochrana pomocí hesel statická metoda pro určení autenticity. Přirozeně může počítač nutit uživatele k pravidelné obměně hesel, např. každý měsíc.

To je ale jen malý pokrok: ***Stejné heslo se vždy použije vícekrát***. Proč je to problém? Mr. X jednou odposlechne přenos hesla a od té chvíle se může vydávat za uživatele.

Typy identifikace XI - Čipové karty

Abychom vyřešili tento problém, musí se data, která jsou vyměňována mezi počítačem a uživatelem neustále měnit – pokaždé nová otázka a nová odpověď.

Je jasné, že uživatelova odpověď se musí formulovat tak, že nikdo jiný nemůže tuto odpověď poskytnout. To znamená, že reakce uživatele na otázku musí záviset na nějakém tajemství, jinak by totiž na ni mohl odpovědět i Mr. X.

Odpovídající protokol podle metody **Challenge-and-Response** probíhá následovně.

Jak počítač tak uživatel mají k dispozici jednosměrnou funkci f a společný tajný klíč K. Počítač obdrží od uživatele data k identifikaci; v našem případě třeba jméno uživatele A. Nato si počítač opatří pro dané jméno příslušný klíč K.

Typy identifikace XII - Čipové karty

Klíč K může být uložen v seznamu, může být odvozen systémovým globálním klíčem nebo dán k dispozici pomocí podobné procedury.

Počítač se musí přesvědčit o tom, že uživatel, který se chce identifikovat, je skutečně uživatel A a ne zákeřný Mr. X.

Pokud může uživatel dokázat, že má správný klíč K, je automaticky uznán za autentického. Tj. považujeme za dokázané, že se skutečně jedná o uživatele A.

Cílem počítače v následující popsané proceduře autenticity je "***nepřímo***" se přesvědčit, že uživatel je ve vlastnictví klíče K. V popisu protokolu označme klíč, který je ve vlastnictví uživatele (to by mohl být i Mr. X) jako K'.

Typy identifikace XII - Čipové karty

Za tímto účelem klade počítač uživateli otázku tím, že mu zašle náhodně zvolené číslo *RAND*.

Pak uživatel vypočte **parametr autenticity** $AP' = f_{K'}(RAND)$ a zašle jej počítači.

Současně počítač vypočte hodnotu $AP = f_K(RAND)$ a porovná zda $AP = AP'$.

Pokud ne, pak něco nesouhlasí, v opačném případě má počítač s vysokou pravděpodobností stejný klíč jako uživatel. Tedy uživatel bude akceptován.

Tato procedura má dvě výhody a dvě nevýhody. Protože je číslo *RAND* náhodně voleno, mění se případ od případu a Mr. X nemá žádnou šanci předpovědět následující *RAND*. Také *AP* má charakter náhodného čísla, takže Mr. X nikdy nemůže na otázku dát správnou odpověď.

Úvod
Integrita a autenticita
One-way
Uživatel
Typy identifikace
Zero-Knowledge

Typy identifikace XII - Čipové karty

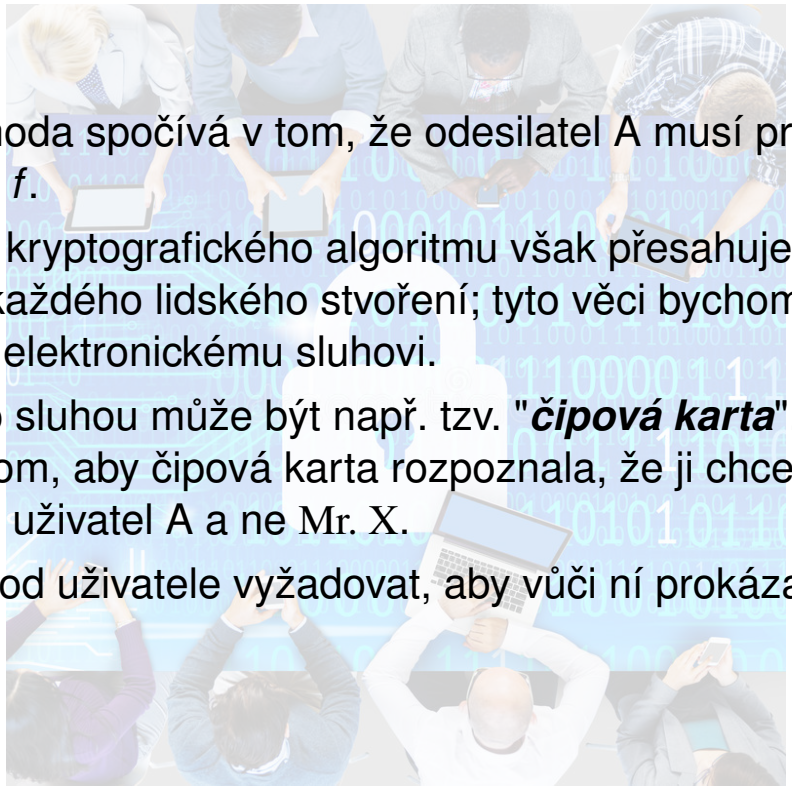
Právě tak důležitá je skutečnost, že pomocí protokolu se prokáže, že oba klíče jsou identické, ale klíče samotné se uvnitř protokolu nikdy neobjeví.

Nevýhody Challenge-and-Response protokolu jsou sice "***malé***", ale nesmějí být jak prakticky tak teoreticky zanedbány.

Uvědomme si, že počítač vlastní přístup k tajným klíčům uživatelů. To počítači umožňuje, aby se vydával vzhledem k jiným instancím (nebo vůči sám sobě) za uživatele.

Jinak řečeno: Předpoklad pro použití Challenge-and-Response protokolu je, že ***uživatelé důvěřují integritě počítače provádějícího autenticitu***.

Typy identifikace XIII - Čipové karty



Jiná nevýhoda spočívá v tom, že odesilatel A musí provést algoritmus f.

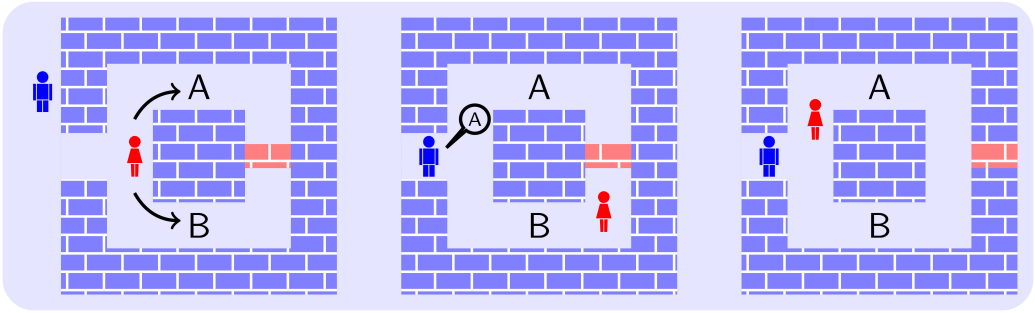
Provedení kryptografického algoritmu však přesahuje znalosti a možnosti každého lidského stvoření; tyto věci bychom měli přenechat elektronickému sluhovi.

Takovýmto sluhou může být např. tzv. "***čipová karta***". Problém je však v tom, aby čipová karta rozpoznala, že ji chce použít oprávněný uživatel A a ne Mr. X.

Musí tedy od uživatele vyžadovat, aby vůči ní prokázal totožnost.

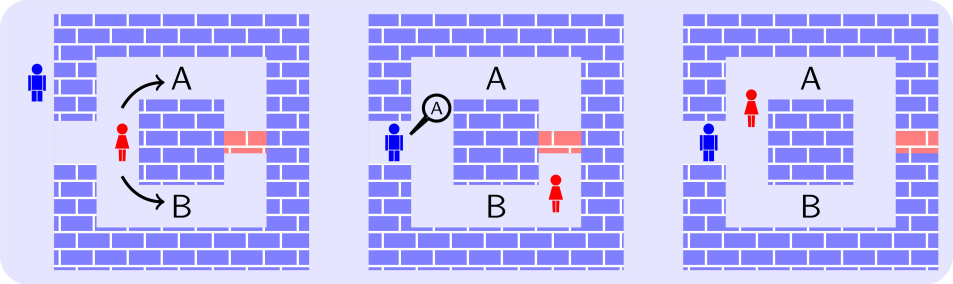
Ilustrativní příklad I

Alice zná tajné slovo, které otevírá dveře v jeskyni. Jak o tom může Boba přesvědčit, aniž by toto tajné slovo odhalila?



Nejprve Bob čeká před jeskyní, když Alice vchází dovnitř.
Označení levé a pravé cesty od vchodu bude A a B.
Alice jde náhodně cestou A nebo B. Poté Bob vstoupí do jeskyně a křičí jméno cesty, kudy chce, aby se vrátila, zvolené náhodně. Pokud opravdu zná kouzelné slovo, je to snadné: v případě potřeby otevře dveře a vrací se podél požadované cesty. Všimněte si, že Bob neví, jakou cestou původně šla.

Ilustrativní příklad II



Předpokládejme však, že Alice toto slovo neznala. Pak by se mohla vrátit po jmenované cestě, pokud by Bob ohlásil jméno stejné cesty, jakou vstoupila. Vzhledem k tomu, že by si Bob náhodně vybral A nebo B, měla by 50% šanci správně hádat. Pokud by tento postup opakovali mnohokrát, řekněte 20krát v řadě, její šance na úspěšné předvídání všech Bobových žádostí by se stala velmi malá.
Z tohoto důvodu, pokud se Alice pokaždé objeví na správném výstupu z jeskyně, může Bob dojít k závěru, že je velmi pravděpodobné, že Alice zná tajné slovo.

Zero-Knowledge protokol I

Zero-Knowledge protokol je procedura, která jedné straně dovolí přesvědčit druhou stranu, že má určité tajemství, aniž by prozradila o tomto tajemství nejmenší informaci.

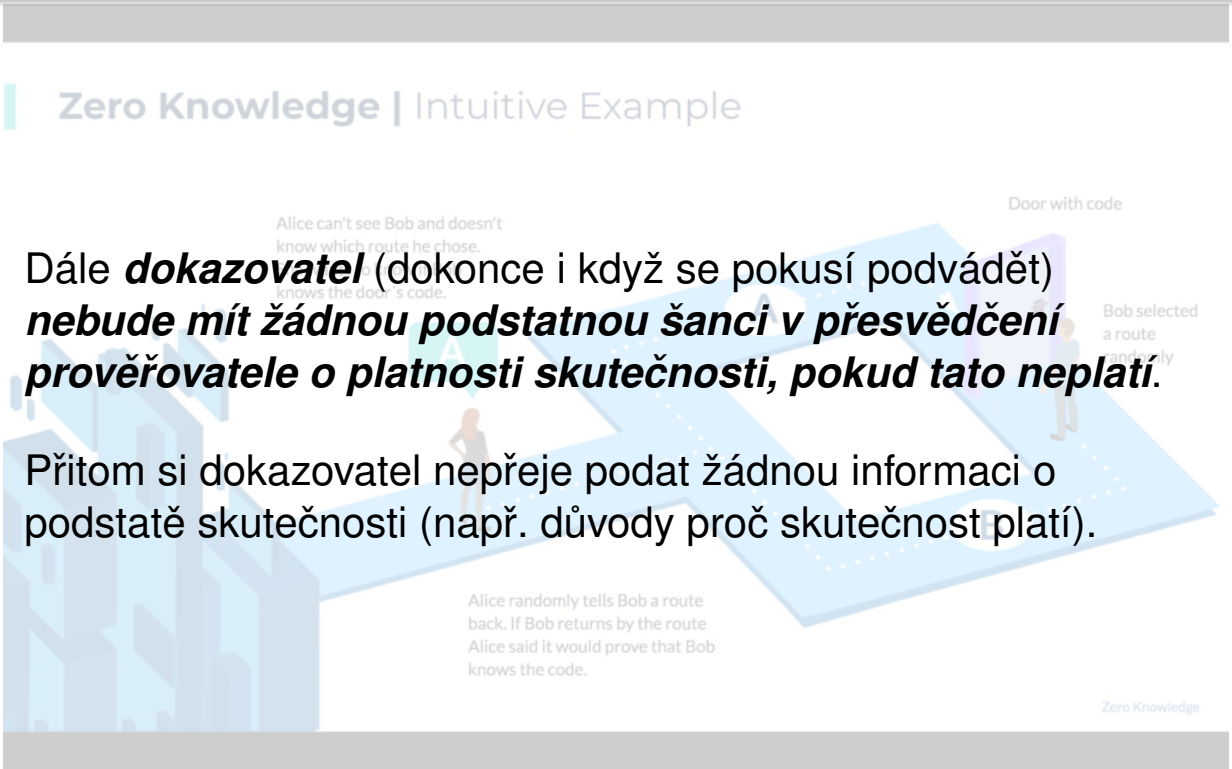
Zero-knowledge protokol je dvoustranný protokol; jedna strana se nazývá ***dokazovatel***, druhá ***prověřovatel.***

Dokazovatel zná nějakou skutečnost a přeje si přesvědčit prověřovatele o této skutečnosti.

Prověřovatel chce protokol, který mu dovolí se přesvědčit o platnosti této skutečnosti právě tehdy, když je tato skutečnost pravdivá.

Přesněji, ***dokazovatel*** (jestliže jedná podle protokolu) ***bude schopen přesvědčit prověřovatele o platnosti skutečnosti, pokud je tato skutečnost pravdivá***.

Zero-Knowledge protokol II



Dále ***dokazovatel*** (dokonce i když se pokusí podvádět) ***nebude mít žádnou podstatnou šanci v přesvědčení prověřovatele o platnosti skutečnosti, pokud tato neplatí***.

Přitom si dokazovatel nepřeje podat žádnou informaci o podstatě skutečnosti (např. důvody proč skutečnost platí).

Zero-Knowledge protokol II - Tartagliovo tajemství

Řešení rovnic byl v historii matematiky nanejvýš důležitý problém. Jednoduché bylo řešení lineárních a kvadratických rovnic. Kubická rovnice, tedy rovnice tvaru

$$ax^3 + bx^2 + cx + d = 0$$

nebyla tak snadno řešitelná a nalezení jejího řešení trvalo podstatně déle.

Benátský matematik Niccolo **Tartaglia** (1499-1557) objevil podle svých údajů v roce 1535 metodu jejího řešení. Veřejně oznámil svůj objev, ale vzorec pečlivě tajil. Tartaglia snadno ukázal, že je schopen kubickou rovnici řešit a tím každého bez obtíží přesvědčil, že jeho řešení je správné.

Zero-Knowledge protokol III - Tartagliovo tajemství

Ačkoliv se Tartagliovi konkurenti mimořádně snažili, nebyli schopni odhalit jeho tajemství.

Až Geronimo **Cardano** (1501-1576) přesvědčil Tartagliu, aby mu vzorec ukázal. Slíbil, že ho udrží v absolutní tajnosti.

Stalo se, co se stát muselo: Ve své knize *Ars Magma* (1545) zveřejnil Cardano Tartagliův vzorec; čestně však popsal, že tato formule pochází od Tartaglii.

Přesto zůstává ironií osudu, že se dnes vzorec pro řešení kubické rovnice nazývá **Cardanova formule**.

Pro nás je důležité, že Tartaglia měl tajemství (metodu řešení kubických rovnic), které byl schopen utajit a o jehož existenci mohl přesvědčit ostatní.

Zero-Knowledge protokol IV - Odmocninový příklad

Uvažme hru probíhající následovně: Máme hráče A a hráče B.

Hráč A tvrdí, že zná číslo s, které hráč B nezná, a s přitom splňuje, že $s^2 \equiv 34 \bmod 55$.

Hráč A chce hráče B přesvědčit, že toto číslo opravdu zná. Proto náhodně zvolí číslo r, umocní je na druhou a spočítá zbytek po dělení 55, řekněme např. 26 a ukáže jej hráči B.

Nyní přichází na řadu hráč B. Hodí si mincí; pokud padne hlava, chce znát od prvního hráče $r \cdot s \bmod 55$, jinak pouze r.

Předpokládejme, že padla hlava. Pak řeknu, že $r \cdot s \bmod 55 = 53$, a to lze jednoduše prověřit, neboť skutečně platí

$$53^2 \bmod 55 = 4 = (26 \cdot 34) \bmod 55 = r^2 \cdot s^2 \bmod 55.$$

Zero-Knowledge protokol V - Odmocninový příklad

Uvědomme si, že je právě tak těžké najít s, pokud navíc známe $r^2 \bmod 55$ a $r \cdot s \bmod 55$, jako kdybychom je obě neznali.

Připomeňme, že r nesmí být zcela náhodně zvoleno; například by r mělo být větší než $\sqrt{55}$.

Lze u této hry podvádět? Ano - v případě, že hráč B neobjeví podvod. Ale po čase se na ***alespoň jeden podvod*** přijde.

Představme si, že hráč A podváděl a číslo s nezná. Kdyby věděl, že mince hráče B neukáže hlavu, neměl by žádný problém ukázat hráči B číslo r. Avšak A by nevěděl, co udělat, kdyby hráč B chtěl znát $r \cdot s \bmod 55$!

Kdyby hráč A věděl, že mince hráče B ukáže hlavu, mohl by A také podvádět; nejprve by si vybral číslo, o kterém by věděl, že jeho zbytek po dělení 55 je čtverec, např $2^2 = 4$ a zvolil by pak

$$r^2 = (4/s^2) \bmod 55 = (4/34) \bmod 55 = 26.$$

Zero-Knowledge protokol VI - Odmocninový příklad

V případě, že by hráč B chtěl znát $r \cdot s \bmod 55$, odpoví jednoduše 2 a hráč B se může přesvědčit, že platí

$$2^2 = (26 \cdot 34) \bmod 55 = (4/34) \cdot 34 \bmod 55.$$

Když ale hráč A musí říci číslo r, okamžitě se přijde ne jeho podvod.

Celkem tedy můžeme říci, že hráč A má v každém kole 50%-ní šanci úspěšného podvodu. Aby se tedy hráč B přesvědčil, že hráč A nelže, musela by se hra hrát více kol.

Tato hra ukazuje ***rozhodující vlastnosti skutečného Zero-Knowledge protokolu***:

Zero-Knowledge protokol VII - Vlastnosti

- Protokol je **interaktivní**; oba partneři provádí náhodné volby (hráč A volí náhodné číslo r, hráč B se rozhodne, zda chce znát r nebo $r \cdot s \bmod 55$).
- ***Pravděpodobnost úspěšného podvodu závisí na počtu odehrátých kol***. V každém kole se pravděpodobnost zmenší na polovičku.

Zero-Knowledge protokol VII - Obarvení grafu

Uvažme následující hru dvou hráčů A a B. Hráč A chce přesvědčit hráče B, že jistý graf je obarvitelný třemi barvami, aniž by mu prozradil konkrétní obarvení.

Přitom hráč A to může provést v posloupnosti $|E|^2$ stavů zadaných následovně:

- Hráč A náhodně přebarví tyto tři barvy (např. všechny červené uzly na modro, všechny žluté uzly na červeno a všechny modré uzly na žluto).
- Hráč A zašifruje barvu každého uzlu pomocí použití šifrovacího schématu s různou pravděpodobností pro každý vrchol. Potom ukáže hráči B všechna tato zašifrování spolu s předpisem přiřazujícím zašifrování s odpovídajícím vrcholem.

Zero-Knowledge protokol VIII - Obarvení grafu

- Hráč B vybere hranu grafu.
- Hráč A provede dešifrování barev dvou uzlů této hrany odkrytím odpovídajících šifrovacích klíčů.
- Hráč B potvrdí, že dešifrování bylo provedeno správně a že dva koncové uzly hrany jsou obarveny dvěma různými, ale legálními hranami.

Je-li graf skutečně 3-barevný (a hráč A zná obarvení), pak hráč B nikdy nezjistí žádnou špatně obarvenou hranu.

V případě, že graf není 3-barevný, pak existuje šance alespoň $|E|^{-1}$ v každém stavu, že hráč A se hráče B pokouší podvést.

Šance, že by hráč A mohl hráče B podvést v $|E|^2$ krocích, je exponenciálně malá.

Zero-Knowledge protokol IX - Obarvení grafu

Poznamenejme, že historie naší komunikace – v případě, že graf je 3-barevný – sestává se zřetězení zpráv odeslaných během každého stavu.

Je možné dokázat, (za předpokladu, že je možné perfektní zašifrování), že pravděpodobnostní rozdělení definované nad těmito průběhy naší množinou možných interakcí je nerozeznatelné v polynomiálním čase od rozdělení, které můžete vytvořit nad těmito průběhy samotnými, bez účasti hráče A.

To znamená, že hráč B nezíská jinou vědomost z protokolu, než že graf je skutečně 3-barevný.

Zero-Knowledge protokol X - Fiat-Shamirův protokol

V roce 1986 předložili izraelští matematici Adi **Shamir** a Amos **Fiat** protokol, který otevřel nové rozměry v určování autenticity uživatele.

Přesněji, jedná se o určení autenticity typu počítač – počítač, přičemž jedním z počítačů je čipová karta uživatele.

Fiat-Shamirův protokol je založen na výsledcích Shafi **Goldwassera** a Silvia **Micaliho** z Massachusettského technologického institutu a rovněž Charles **Rackoffa** z Univerzity Toronto.

Verze, kterou nyní popíšeme, je zobecněním odmocninového příkladu.
Bezpečnost protokolu spočívá na tom, že ***je mimořádně obtížné najít druhou odmocninu nějakého čísla*** v ***modulo*** n***.***

Zero-Knowledge protokol XI - Fiat-Shamirův protokol

Přesněji, buď dáno přirozené číslo n, které není prvočíslo. ***Jestliže neznáme rozklad*** n ***na součin prvočísel, je prakticky nemožné najít číslo*** s ***tak, že*** $s^2 \bmod n = v$.

Doporučeje se volit n tak veliké, aby bylo řádově asi 10^{200}; to znamená, že normálním písmen je n dlouhé asi 1 m.
Před vlastním procesem určení autenticity uživatele pomocí **Fiat-Shamirova protokolu** se zvolí v šifrovací centrále dvě prvočísla p a q a utvoří se jejich součin $n = p \cdot q$.

Je rozhodující, že centrála drží p a q v tajnosti, zatímco n je veřejně známo. Proto musí být n tak veliké, že faktorizace n je odsouzena k neúspěchu.

Důvod proto je, že můžeme relativně jednoduše určit druhé odmocniny z mod p a mod q. Z těchto čísel lze pak snadno získat druhou odmocninu mod n.

Zero-Knowledge protokol XII - Fiat-Shamirův protokol

Centrála spočítá pro každého uživatele číslo s (které je tajemství uživatele) a číslo v tak, že platí $v = s^2 \bmod n$. Číslo v slouží k veřejné identifikaci uživatele. Centrála by mohla zvolit pro v identifikační údaje uživatele (v binární podobě) a odtud vypočítat s. Číslo n je veřejná systémová konstanta.

Tímto jsou už všechny úlohy centrály popsány. Zejména už centrála nehraje při aktuálních procesech určení autenticity žádnou úlohu. Vlastnosti protokolu jsou:

- Oba počítače musí provést jen několik málo výpočtů **mod** n: na straně uživatele se musí pouze umocnit na druhou náhodné číslo r; v polovině všech případů se musí ještě spočítat $r \cdot s$. Počítač provádějící určení autenticity uživatele musí umocnit na druhou číslo y a v polovině všech případů vynásobit x s v.

Zero-Knowledge protokol XIII - Fiat-Shamirův protokol

- Počítač používá pouze veřejně dostupné informace, zatímco uživatel podstatným způsobem používá tajemství znalosti s.
- Počítač se za jistou dobu přesvědčí, že uživatel zná opravdu tajemství s. Pravděpodobnost, že by nepřítel, který nezná tajemství s, pokaždé předpověděl, který bit b bude zvolen, je při t-násobném opakování protokolu jen $1/2^t$. V případě, že $t = 20$, je tato pravděpodobnost méně než 1 ku miliónu.
- I po provedení určení autenticity uživatele zůstane tajemství s absolutně utajeno.
- Bezpečnost protokolu je v rozhodné míře závislá na tom, že výpočet odmocnin **mod** n je tak obtížné, že ani sám Mr. X není schopen vypočítat druhou odmocninu z v – a to ani tehdy ne, když odposlechne tisíce transakcí mezi uživatelem a počítačem.