UNIVERZITA PARDUBICE

Fakulta ekonomicko - správní

Autentizační protokoly

Ladislav Bareš

Diplomová práce

2010

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ladislav BAREŠ**
Osobní číslo:	**E08433**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Autentizační protokoly**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Tato práce si klade za cíl analyzovat současný stav v oblasti autentizačních protokolů. Zároveň vytváří vzorovou e-learningovou učební jednotku, která demonstruje jednotlivé přístupy.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

STALLING, William. Operating Systems. Internals and Design Principles. 5th edition. [s.l.] : Prentice Hall, 2004. 832 s. ISBN 978-0131479548.
SILBERSCHATZ, Abraham, GALVIN, Peter Baer, GAGNE, Greg. Operating System Concepts. 7th edition. [s.l.] : Wiley, 2004. 944 s. ISBN 978-0471694663.
TANENBAUM, Andrew S. Modern Operating Systems. 3rd edition. [s.l.]: Prentice Hall, 2007. 1104 s. ISBN 978-0136006633.
Internetové zdroje.

Vedoucí diplomové práce: **prof. Ing. Jan Čapek, CSc.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **5. října 2009**

Termín odevzdání diplomové práce: **30. dubna 2010**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 5. října 2009

PROHLÁŠENÍ

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 30. 4. 2010

Ladislav Bareš

PODĚKOVÁNÍ

prof. Ing. Janu Čapkovi, CSc. za vedení při tvorbě diplomové práce a poskytnutí materiálů.

Celé FES Univerzity Pardubice za vytvoření přátelské atmosféry při studiu.

ANOTACE

Tato práce se zabývá tématem autentizace a autentizačních protokolů. Vysvětluje jejich základní principy, způsob fungování, přičemž důraz je kladen na autentizaci typu „něco znát“. Zároveň obsahuje vzorovou učební jednotku, vytvořenou a určenou pro online vzdělávací prostředí Moodle.

KLÍČOVÁ SLOVA

Autentizace, autentizační protokoly, NTLM, Kerberos, hesla, biometrika

TITLE

Authentication protocols

ANNOTATION

This thesis deals with the subject of authentication and authentication protocols. It explains their basic principles and ways of functioning, whilst in particular stresses the authentication of the so-called know-something type. At the same time it consists of a sample learning unit created and intended for Moodle - the online learning environment.

KEYWORDS

Authentication, authentication protocols, NTLM, Kerberos, passwords, biometrics

OBSAH

Seznam zkratek:

AES – Advanced Encryption Standard
AS – Authentication Service
CHAP - Challenge Handshake Authentication Protocol
CIA - Confidentiality Integrity Availability
DC – Domain Controller
DES – Data Encryption Standard
DoS – Denial Of Service
EER – Equal Error Rate
FAR – False Accept Rate
FRR – False Reject Rate
FQDN – Fully Qualified Domain Name
KDC – Key Distribution Center
LM – LAN Manager
LMOWF – LAN Manager One Way Function
LSASS – Local Security Authority SubSystem
MAC – Message Authentication Code
NTLM - NT LAN Manager
OWF – One Way Function
PAP – Password Authentication Protocol
RSA – Rivest, Shamir, Adelman (jména autorů)
SAM - Security Accounts Manager
SAS – Secure Attention Sequence
TGS – Ticket Granting Service
TGT – Ticket Granting Ticket

1 Úvod

Tato práce má za cíl poukázat na základní požadavky autentizačních protokolů informačních a operačních systémů, přičemž za protokol pro účely této práce považujeme nějaký předem daný postup, který se řídí stanovenými pravidly, které říkají co, kdy a jak bude probíhat. V obecné části je nejprve nastíněna problematika šifrování a autentizačních principů z hlediska zasílání zpráv. Důraz v této práci je ovšem kladen na autentizaci typu „něco znát“ a přihlášení uživatele, ať již v obecné rovině, kde je tato problematika vysvětlena na protokolu autorů Hölbl, Welzer, Brumen, případně v rovině konkrétní platformy Microsoft Windows, u nichž jsou vybrány důležité aspekty implementací jí používaných autentizačních protokolů.

Výstupem této práce je dále vzorová učební jednotka na téma autentizace a autentizační protokoly, která byla zpracována v online vzdělávacím prostředí Moodle.

2 Bezpečnost informačních systémů

V úvodu této práce budou rozebrány základní principy bezpečnosti, které jsou kladeny na informační a operační systém jako celek, přičemž budou zmíněny i vybrané typy útoků na tyto principy.

2.1 Úrovně bezpečnosti

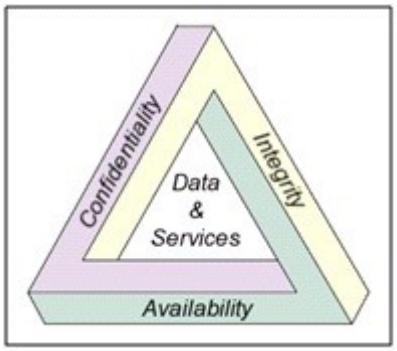
Pro ochránění systému, je třeba budovat bezpečnost ve čtyřech úrovních [25]:

- Fyzická – místo, kde se nachází počítačové systémy, musí být fyzicky zabezpečeno proti případným vniknutím nepovolané osoby.
- Lidská – autentizace uživatelů musí být prováděna důsledně, aby pouze oprávněný uživatel mohl přistupovat k systému. Je třeba minimalizovat útoky typu sociálního inženýrství, jakými jsou phishing (legitimně se tvářící email případně webová stránka oklame uživatele a ten vloží své přihlašovací údaje), dále pak metoda „dumpster diving“, která spočívá v získání informací pro přístup k systému pomocí hledání příslušných informací v telefonních seznamech, poznámkách uživatelů systému, jejich koších apod.
- Bezpečnost operačního systému – systém sám by měl být odolný proti metodám útoku.
- Počítačová síť – mnohá data dnes proudí přes soukromé, ale i přes veřejné, otevřené kanály (internet). Zachycení těchto dat může napáchat stejně takové škody jako samotný útok na informační systém.

Pokud chceme zajistit bezpečnost operačního systému, je nutné, aby byla důsledně dodržována bezpečnost na prvních dvou úrovních, neboť jejich nedokonalá zabezpečení mohou vést k jednoduššímu obcházení bezpečnostních pravidel na úrovni operačního systému.

2.2 Princip CIA

Confidentiality-Integrity-Availability (CIA) je široce užívaný test pro vyhodnocení bezpečnosti informačních systémů a proto jej lze nalézt i v požadavcích na autentizační protokoly, které by tyto tři základní principy měly podporovat.



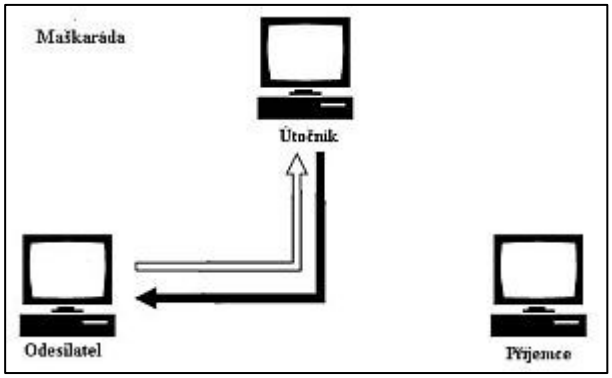
Obrázek 1 - Princip CIA [30]

Z výše uvedeného lze dovodit, že těmito principy jsou princip důvěrnosti, integrity a dostupnosti.

Z hlediska bezpečnostních problémů tedy můžeme identifikovat útoky a události, které bezprostředně napadají jeden z uvedených principů, přičemž tyto problémy rozeznáváme dvojího druhu [25]: úmyslné a náhodné. Je zřejmé, že je mnohem jednodušší bránit se problémům kategorie náhodný. Níže je uveden neúplný výčet možných útoků a událostí na informační systém (pod pojmem útok chápeme pokus o překonání bezpečnostních opatření):

- porušení důvěrnosti – tento typ události zahrnuje neautorizovaný přístup k datům. Toto je převážný cíl útočníků, neboť získání citlivých dat ze systému, například přihlašovací údaje uživatelů, může v konečném důsledku vést k peněžnímu prospěchu pro daného narušitele.
- porušení integrity – tento typ události zahrnuje neautorizovanou modifikaci dat. To znamená, že útočník zmodifikuje nějakou část informačního systému, její zdrojový kód apod.
- porušení dostupnosti – neboli neautorizované zničení dat. V případě, že útočník pouze znepřístupnil tato data, jedná se o útok typu DoS.
- „theft of service“ - tento typ události zahrnuje neautorizované používání zdrojů. Například může útočník do systému nainstalovat službu, která se bude navenek tvářit jako souborový server.
- „stolen-verifier“ útok – jedná se o útok, kdy útočník získá autentikátor (zahashované heslo uživatele např.), který je uchován na serveru a tento autentikátor použije v procesu ověřování identity uživatele.
- „Denial-Of-Service“ (DoS útok) – znepřístupnění služby příliš velkým počtem požadavků na službu, případně způsobení, že se služba nechová takovým způsobem, jakým by měla.

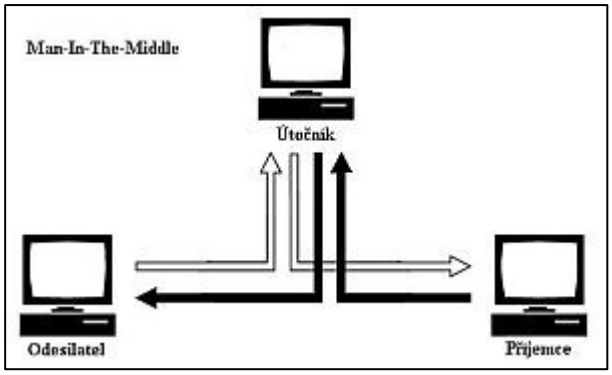
Abychom mohli přecházet výše zmíněným událostem, je nutné zmínit techniky, které útočníci ve snaze vyvolat některou z událostí používají [25]. Nejčastější metodou je metoda tzv. **maškarády**. Jedná se o metodu, kdy jedna z komunikujících stran se vydává za někoho jiného (jinou osobu, server). Použitím této techniky útočník překoná autentizační mechanismy, jež zaručují identifikaci komunikujících stran. Útočník tak získá práva a přístup do systému, který by za normálních okolností nezískal.



Obrázek 2 - Metoda maškarády [upraven ze 7]

Dalším z běžných útoků je útočníkova odpověď na zachycenou výměnu dat. Někdy tento útok předchází vlastnímu útoku na data. V informačních systémech se ale převážně jedná o útok, který modifikuje zprávu tak, aby útočník opět zvýšil svoje uživatelská práva v systému.

Jinou metodou využívanou útočníky je metoda **man-in-the-middle**, při které útočník figuruje jako prostředník při probíhající komunikaci. Pro odesilatele zprávy se tváří jako příjemce a naopak, aniž by účastníci komunikace o tomto věděli.



Obrázek 3 - Metoda man-in-the-middle [upraven ze 7]

3 Autentizace – všeobecný pohled a její zajištění

Autentizace je proces ověření proklamované identity subjektu [25]. Jinými slovy daný subjekt se přihlašuje do systému a tento jej na základě příslušné metody autentizuje, tedy potvrdí, že je oprávněn využít zdroj, ke kterému se přihlašuje. Toto se děje ve dvou krocích [27]:

- Identifikační krok – uživatel sděluje systému, ke kterému se přihlašuje, svůj identifikátor.
- Verifikační krok – uživatel poskytuje systému svou autentizační informaci, která potvrdí vazbu mezi ním a jeho identifikátorem.

Pokud je daný systém izolován, operační systém má plnou kontrolu nad komunikací, která probíhá, neboť ovládá veškeré komunikační kanály. Naproti tomu v počítačové síti je tomu jinak. V síťovém prostředí počítače obdrží pouze shluk bitů, přičemž nemají možnost jak zjistit, který stroj, případně aplikace tuto zprávu odeslala. Obdobně, odesilatel zprávy (počítač) vyšle zprávu do síťového prostředí s tím, že neví, která ze stanic danou zprávu přijme.

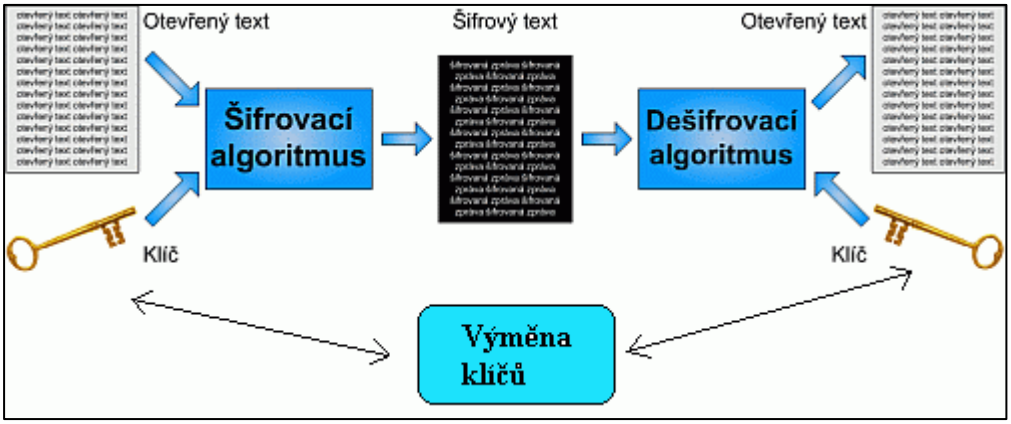
Běžně se používá IP adresa pro určení obou účastníků komunikace. Tedy při obdržení packetu, ten obsahuje svou zdrojovou adresu a v případě, že počítač vysílá zprávu, identifikuje příjemce pomocí jeho IP adresy. Pokud se spolehneme pouze na toto, lze si velmi dobře představit situaci, kdy útočník podvrhne zdrojovou adresu a stejně tak i ostatní stanice, jiné než specifikované cílovou adresou, mohou mít přístup k takto vyslanému paketu (např. routery). Jak tedy operační systém rozhodne, komu umožní přístup ke sdíleným prostředkům, když nemůže důvěřovat zdroji pouze na základě IP adresy? Stejně tak jak zajistit, aby pouze příjemci byla odesílaná data srozumitelná a ostatním byl obsah těchto dat skryt?

3.1 Kryptografie

je věda, která studuje šifrovací algoritmy, kryptografické nástroje, hardwarové implementace šifrovacích algoritmů, kryptografické protokoly apod. [5]. Moderní kryptografie je založena na šifrování pomocí klíčů, které se distribuují účastníkům komunikace. Kryptografie tak ve své podstatě dovoluje příjemci zprávy ověřit, že příslušná zpráva byla vytvořena odesilatelem vlastnícím příslušný klíč. Naproti tomu odesilatel může svou zprávu zašifrovat tak, že pouze příjemce zprávy bude schopen tuto zprávu dešifrovat.

3.1.1 Šifrování

Na obrázku níže je uvedeno, jak probíhá zabezpečená komunikace přes nezabezpečený kanál. Výměna klíčů mezi zúčastněnými stranami může proběhnout buď přímo, nebo přes důvěryhodnou třetí stranu tzv. certifikační autoritu.



Obrázek 4 - Základní schéma šifrování [upraven z 18]

Šifrovací algoritmus se skládá z následujících komponent [25]:

- množina klíčů *K*
- množina zpráv *M*
- množina šifrovaných zpráv *C*
- funkce $E: K \rightarrow (M \rightarrow C)$ taková, že pro každé *k* z množiny *K*, *E(k)* je funkce, která generuje šifrovanou zprávu z otevřené zprávy. *E* i *E(k)* pro jakékoliv *k* by měli být vypočitatelné funkce.
- funkce $D: K \rightarrow (C \rightarrow M)$ taková, že pro každé *k* z množiny *K*, *D(k)* je funkce, která generuje otevřenou zprávu ze šifrované zprávy. *D* i *D(k)* pro jakékoliv *k* by měli být vypočitatelné funkce.

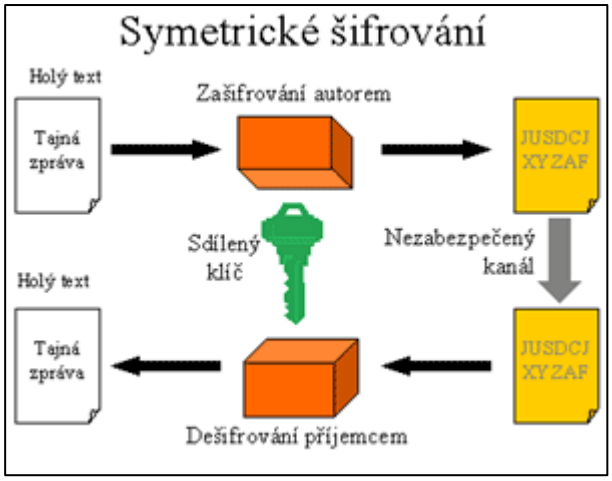
Pro šifrovací algoritmus musí být splněna základní podmínka: Existuje šifrovaný text *c* z množiny *C*, potom počítač může vypočítat otevřený text *m* tak, že $E(k)(m) = c$ pouze tehdy, pokud zná (ovládá) *D(k)*. Jinými slovy pouze a jen počítač, který zná *D(k)* může zprávu dešifrovat. Jelikož jsou zašifrované zprávy zasílány prostřednictvím otevřených kanálů (Internet) a jsou tak komukoliv přístupné, je důležité, aby bylo výpočetně nemožné odvodit *D(k)* ze zašifrované zprávy. Rozlišujeme dva druhy šifrovacích algoritmů: symetrické a asymetrické.

3.1.2 Symetrické šifry

Při použití symetrického šifrování je použit pro zašifrování a dešifrování stejný klíč. V tomto případě se tedy *E(k)* dá odvodit z *D(k)* a opačně.

Výhodou symetrických šifer je jejich nízká výpočetní náročnost. Na druhou stranu velkou nevýhodou je nutnost sdílení tajného klíče, takže se odesilatel a příjemce zprávy musí předem domluvit na tajném klíči, což může být někdy problém, případně si je vyměnit například

algoritmem Diffie–Hellman uvedeným v kapitole 9.1. Neznámějšími zástupci tohoto způsobu šifrování je DES, 3DES, AES.

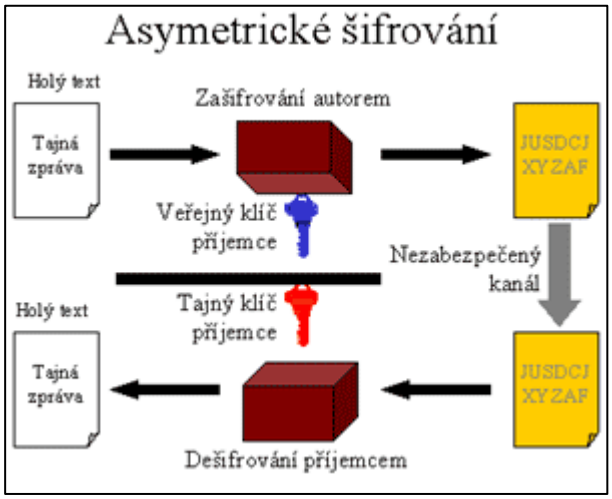


Obrázek 5 - Schéma symetrického šifrování [26]

3.1.3 Asymetrické šifry

Při asymetrické šifře se k šifrování a dešifrování používá jiný klíč. Výhodou tohoto řešení je, že příjemce nemusí s odesilatelem sdílet žádné tajemství (šifrovací klíč), čímž eliminuje potřebu výměny klíčů mezi komunikujícími stranami.

Vlastní šifrování začíná tvorbou dvojice klíčů, které jsou spolu matematicky svázány a dále publikací veřejného klíče (k_e), přičemž privátní (tajný) klíč (k_d) je velmi pečlivě střežen, neboť kdokoliv s přístupem k tomuto klíči může dešifrovat zprávu zašifrovanou odpovídajícím veřejným klíčem.



Obrázek 6 - Schéma asymetrického šifrování [26]

Asymetrická kryptografie je založena na matematických zákonitostech a je tak výpočetně mnohem náročnější, než symetrické šifrování, což ji činí dražší. I když se tato metoda nepoužívá na šifrování zpráv velkého objemu, je využitelná při autentizaci, důvěrnosti a distribuci klíčů.

3.2 Autentizační schéma

Jak bylo zmíněno, autentizace je proces ověření proklamované identity subjektu. Na základě výše uvedeného lze tvrdit, že autentizace se co do funkčnosti překrývá se šifrováním (asymetrickým), neboť je pro splnění svého účelu využívá [25]. Na zašifrovanou zprávu tak můžeme pohlédnout i tak, že nejenom že sděluje nějakou informaci, ale zároveň i potvrzuje identitu odesilatele. Pokud například $D(k_d, N)(E(k_e, N)(m))$ vygeneruje platnou zprávu, potom lze s určitostí říct, že tvůrce zprávy musel při její tvorbě znát k_e. Autentizace je dále vhodná i pro dokázání, že zpráva nebyla modifikována.

Základní schéma autentizace je následovné [25]:

- množina klíčů K
- množina zpráv M
- množina autentikátorů A
- funkce S$: K \rightarrow (M \rightarrow A)$ taková, že pro každé k z množiny K, S(k) je funkce, která generuje autentikátor z otevřené zprávy. S i $S(k)$ pro jakékoliv k by měli být vypočitatelné funkce.
- funkce V$: K \rightarrow (M \times A \rightarrow \{pravda, nepravda\})$ taková, že pro každé k z množiny K, $V(k)$ je funkce, která verifikuje autentikátor zprávy. V i $V(k)$ pro jakékoliv k by měli být vypočitatelné funkce.

Pro autentizační algoritmus musí být splněna základní podmínka: Pro zprávu m, počítač vygeneruje autentikátor a z množiny A takové, že $V(k)(M,a) = pravda$ pouze a jen tehdy, když zná $S(k)$. Poté tedy počítač vlastnící $S(k)$ může generovat autentikátory zprávy takové, že kterýkoliv jiný počítač, který zná $V(k)$ je může ověřit. Tedy počítač, který nezná $S(k)$, nemůže generovat autentikátory zprávy, které by mohly být verifikovány příslušným $V(k)$. Jelikož jsou autentikátory povětšinou odesílány spolu se zprávou přes nezabezpečený kanál, musí být zaručena nemožnost výpočtu $S(k)$ z autentikátoru.

Stejně tak jako existují dva druhy šifrovacích algoritmů, existují i dva druhy autentizačních algoritmů. Pro jejich pochopení je nutné se nejprve seznámit s pojmem hashovací funkce.

3.2.1 Hash

Hash(ovací) funkce je transformace, která jako vstup přijímá řetězec znaků o libovolné délce a výsledkem je pak řetězec znaků s pevnou délkou, tzv. *hash* nebo také *otisk*.

Mezi hlavní vlastnosti této funkce patří [6]:

- jakékoliv množství vstupních dat poskytuje stejně dlouhý výstup (otisk),
- malou změnou vstupních dat dosáhneme velkou změnu na výstupu (tj. výsledný otisk se od původního zásadně na první pohled liší),
- z hashe je prakticky nemožné rekonstruovat původní text zprávy,
- v praxi je vysoce nepravděpodobné, že dvěma různým zprávám odpovídá stejný hash, jinými slovy pomocí hashe lze v praxi identifikovat právě jednu zprávu (ověřit její správnost).

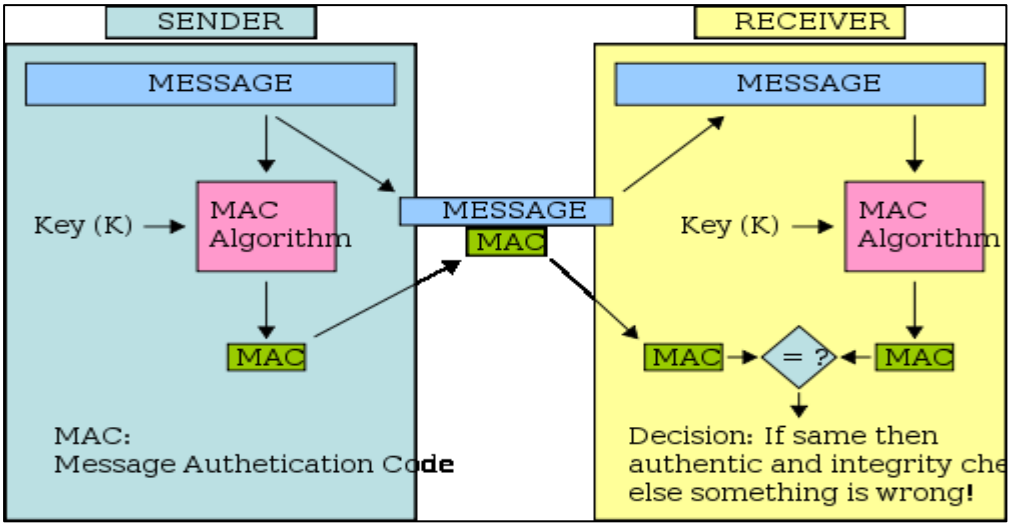
Požadavek na poslední bod lze zapsat matematicky takto:

Existují-li zprávy m_1 a m_2 takové, že $m_1 \neq m_2$ potom $H(m_1) \neq H(m_2)$ pro každou dvojici $m_{i,j}$. Je zřejmé, že pokud $H(m_1) = H(m_2)$, potom $m_1 = m_2$, což znamená, že zpráva nebyla modifikována.

V dnešní době se pro tyto účely používá např. hashovací algoritmus SHA-1, případně SHA-256.

3.2.2 Symetrické šifrování při autentizaci

Prvním typem autentizačního algoritmu, který využívá symetrické šifrování je algoritmus MAC (Message-Authentication Code) [25], který je popsán na následujícím obrázku:

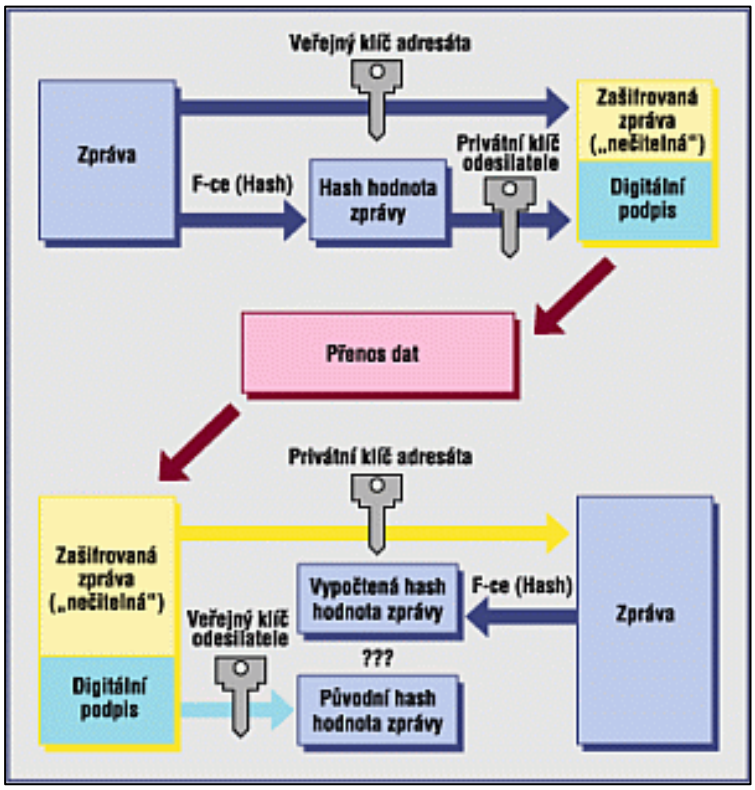


Obrázek 7 - Schéma MAC [9]

Při použití tohoto algoritmu je za pomoci tajného klíče vygenerován kontrolní součet MAC. Znalosti *V(k)* a *S(k)* jsou si ekvivalentní (jedna může být odvozena z druhé), tedy je nutné uchovat *k* v tajnosti. MAC tedy definuje $S(k)(m) = f(k, H(m))$, kde *f* je jednosměrná funkce ve svém prvním argumentu (tedy *k* nelze odvodit z $f(h, H(m))$. A protože je zaručena bezkoliznost hashovací funkce, lze dovodit, že neexistuje zpráva, která by generovala stejný MAC. Pro verifikaci lze následně využít matematický aparát ve tvaru $V(k)(m,a) \equiv (f(k,m) = a)$. Je nutné si ale uvědomit, že *k* je třeba k výpočtu *S(k)* i *V(k)*, tedy že kdokoliv je schopen vypočítat jednu, je schopen vypočítat i druhou.

3.2.3 Asymetrické šifrování při autentizaci

Druhým typem autentizačního algoritmu je algoritmus digitálního podpisu [25]. V tomto algoritmu je výpočetně neproveditelné odvodit $S(k_s)$ z $V(k_v)$, tedy *V* je jednosměrná funkce. Poté tedy k_v je veřejný klíč a k_s je privátní klíč. Tento algoritmus je podobný šifrovacímu algoritmu RSA s tím rozdílem, že použití klíčů je obrácené. Digitální podpis zprávy je vypočítán ze vztahu $S(k_s)(m) = H(m)^k \ mod \ N$. Klíč k_s tvoří pár (d, N), kde *N* je výsledek násobení mezi dvěma náhodně vybranými velkými prvočísly *p* a *q*. Ověření lze provést na základě vztahu $V(k_v)(m,a) \equiv (a^{kv} \ mod \ N = H(m))$, kde k_v splňuje $k_v k_s \ mod \ (p-1)(q-1) = 1$.



Obrázek 8 - Schéma asymetrické autentizace – digitální podpis [19]

Pokud porovnáme šifrovací algoritmy s algoritmy určenými pro autentizaci, najdeme společnou vlastnost. Obě řešení dokáží potvrdit identitu odesilatele, přesto ale jsou využívány algoritmy autentizační, k čemuž vedou tři hlavní důvody [25]:

- Autentizační algoritmy potřebují mnohem méně výpočetního výkonu (pokud odhlédneme od digitálního podpisu na bázi RSA) což se může projevit při zajištění autentizace velké zprávy.
- Autentikátor zprávy je téměř vždy mnohem menší než zpráva samotná a její šifrovaná podoba, což šetří místo a čas při přenosu.
- Někdy vyžadujeme pouze autentičnost zprávy a nikoliv důvěrnost. Jinými slovy nám postačuje „podepsat“, že danou zprávu jsme vytvořili my, ale samotný obsah zprávy již neskrýváme.

Na základě tohoto lze říci, že autentizační algoritmy jsou součástí mnoha bezpečnostních řešení.

4 Autentizace uživatele

V předcházející části byly vysvětleny základní principy autentizace spolu s příklady autentizací zpráv. Z hlediska operačního systému je dále důležitá i autentizace uživatele, neboť pokud není systém schopen autentizovat uživatele, potom autentizovat zprávu od tohoto uživatele je zbytečné [25].

Ve všeobecnosti je autentizace uživatelů založena na jednom, případně všech třech způsobech (v takovém případě hovoříme o trojfaktorové autentizaci). Jedná se o autentizaci typu:

- něco znát (identifikátor, heslo)
- něco mít (klíč, karta)
- něčím být (založena na biometrii – otisk prstů, sítnice)

4.1 Hesla – něco znát

Používání hesel pro autentizaci uživatelů je velice rozšířené. Z hlediska náročnosti na správu a nákladů na jejich zavedení + administraci patří mezi nejzanedbatelnější položky. Zároveň se jedná o velice snadnou metodu, jak uživatel může potvrdit, že on je tím, za koho se vydává [25]. Toto potvrzuje znalostí převážně svého uživatelského jména a zvoleného hesla. Jedná se tedy o autentizaci typu „něco znát“.

Nevýhodou tohoto řešení je ovšem možnost zjištění hesla neoprávněnou osobou. Ta toto heslo může buď odhadnout (na základě informací o osobě), neboť se velice často stává, že uživatelé volí hesla z jejich okolí – jméno za svobodna, datum narození apod. Druhou možností je heslo napadnout tzv. „hrubou silou“ (o útocích na hesla dále v textu kapitoly 4.1.4) neboli vyzkoušením všech možných kombinací znaků, které se mohou v hesle vyskytnout při dané implementaci (malá, velká písmena, interpunkce, číslice) až do nalezení správné kombinace. Proto je velice důležité volit hesla delší, složená z různých skupin znaků pro znesnadnění tohoto úkolu.

Další možností jak získat neoprávněně heslo je jejich vizuální, případně elektronické odchycení. V případě vizuálního zachycení se jedná o útok „přes rameno“, kdy útočník pohledem na klávesnici odpozoruje vkládané heslo. V případě elektronického odchycení se jedná o monitorování sítě a toku dat, které v ní probíhají se zachycením přihlašovacích údajů. Zašifrováním tohoto datového toku může tento problém vyřešit.

Je otázkou příslušné implementace systému, jaké požadavky bude klást na uživatelovo heslo. Uživatel si může heslo zvolit, případně mu jej systém vygeneruje. V obou případech, pokud se bude jednat o nutnost volby hesla v délce desítek znaků, zcela jistě bude bezpečnost vyšší než v případě krátkých hesel, ovšem nároky na zapamatování pro uživatele budou neúměrné, což bude svádět

k napsání tohoto hesla na papír, případně k poznamenání v počítači (zejména v případě hesla generovaného počítačem). Pokud si uživatel heslo volí samostatně, některé systémy kontrolují, zda je toto heslo přijatelné z hlediska bezpečnostní politiky a nabádají uživatele k větší bezpečnosti, případně nutí uživatele měnit heslo v poměrně krátkých intervalech apod., o čemž bude dále pojednáno. Vždy je tedy nutné toto brát v potaz a vyvážit tak roviny pohledu na požadavek „síly“hesel.

4.1.1 Větší zabezpečení – grafická hesla

Hesla jsou v převážné většině textová. Tato hesla jsou velice snadno zranitelná proti útoku „přes rameno“, případně proti keyloggerům, aplikacím, které monitorují stisk kláves. Abychom zajistili vyšší bezpečnost, lze do systému implementovat tzv. grafická hesla.

	Image size	Grid square size (pixels)	Alphabet size/ No. squares	Length/No. click points	Password space size
Alphanumeric	N/A	N/A	64	8	2.8×10^{14}
Alphanumeric	N/A	N/A	72	8	7.2×10^{14}
Alphanumeric	N/A	N/A	96	8	7.2×10^{15}
Graphical	451 × 331	20 × 20	373	5	7.2×10^{12}
Graphical	1024 × 752	20 × 20	1925	5	2.6×10^{16}
Graphical	1024 × 752	14 × 14	3928	5	9.3×10^{17}
Graphical (1/2 screen used)	1024 × 752	14 × 14	1964	5	2.9×10^{16}

Obrázek 9 - Porovnání množiny hesel pro alfanumerická hesla a pro hesla systému PassPoints. [upraven z 32]

Z jednoduchých [3] grafických přihlášení v současnosti převládají grafické klávesnice, tedy zobrazení rozložení jednotlivých kláves a jejich „mačkání“ poklepáním tlačítkem myši. Tato varianta dokáže ochránit před klasickými keyloggery, ovšem nikoliv proti útoku „přes rameno“.

Jiným přístupem je přístup[1], jenž dokáže velice úspěšně zabránit útoku „přes rameno“. První fáze, tedy tvorba hesla, spočívá v zapamatování množiny vybraných obrázků, kdy se nemusí jednat o uspořádanou posloupnost. Ve fázi přihlašování se pak tyto obrázky nahodile rozmístí v přehršli grafických motivů (například tři obrázky tvořící heslo se ztratí v matici 10 x 10). Jakmile uživatel identifikuje svou přihlašovací trojici, musí tlačítkem myši klepnout do pomyslného trojúhelníku, jehož vrcholy tato trojice tvoří.

[1] Vice zde: http://clam.rutgers.edu/~birget/grPssw/

Pro větší bezpečnost je vhodné, aby tyto byly při každém přihlášení náhodně rozmístěny, identifikace se případně několikrát opakovala a vyhraničené oblasti byly co nejmenší. Množina všech možných grafických hesel je větší než u textových, přičemž nároky na zapamatování pro uživatele jsou mnohem nižší, viz Obrázek 9.

4.1.2 Hesla na jedno použití

Pokud abstrahujeme od požadavků správců systémů měnit si uživatelská hesla co nejčastěji (nejlépe pro každé přihlášení jiné heslo), vede nás toto k myšlence tzv. one-time password, tedy jednorázového hesla [29]. Pokud je tento systém použit, uživatel dostane seznam, ve kterém jsou uvedena hesla. Každé přihlášení poté vyžaduje vždy následující heslo v seznamu. Předpokladem této metody je ovšem požadavek na bezpečné uložení seznamu hesel, které nemusí být vždy dobře splnitelné a pro uživatele toto může být obtěžující.

S metodou, která tento požadavek odbourává a zároveň uživateli dovoluje se bezpečně přihlašovat přes nezabezpečený kanál, přičemž útočníkům je dovoleno zachytit veškerý přenos, který protéká oběma směry mezi přihlašovací stanicí a autentizacím serverem, nastínil Leslie Lamport [16]. Tato metoda bývá označována jako „one-way hash chain“, tedy jednosměrné hash zřetězení.

Algoritmus je založen na hashovací funkci, vstup i výstup mohou být stejně dlouhé a spočívá v následujících krocích [16]:

1. Uživatel si zvolí tajný řetězec *s*, který si zapamatuje.
2. Uživatel si dále zvolí tajné číslo *n*, které určuje, kolikrát bude algoritmus generovat jednorázové heslo (jako příklad můžeme zvolit $n = 4$, ačkoliv v praxi se využívají mnohem větší čísla).

Za splnění podmínek uvedených výše bude první heslo vypočítáno hashovací funkcí *f*, která bude použita *n*-krát tzn.:

$$P_1 = f(f(f(f(s))))$$

Druhé heslo bude vypočítáno hashovací funkcí *f*, která bude použita *n-1* krát, tedy:

$$P_2 = f(f(f(s)))$$

Třetí heslo následně spustí hashovací funkci *f* pouze dvakrát, zatímco čtvrté heslo pouze jednou. Ve všeobecnosti lze psát vztah:

$$P_{(i-1)} = f(P_i)$$

Klíčovou vlastností tohoto způsobu výpočtu hesel je fakt, který vychází z podstaty hashovací funkce. Pokud známe nějaké heslo v sekvenci, je velice snadné vypočítat předcházející heslo, ale obtížné spočítat heslo následující. Například, známe-li heslo P_2, potom lehce najdeme heslo P_1, ale nikoliv už P_3.

Na počátku se tedy server inicializuje hodnotou P_0, která je definována jako $f(P_1)$. Hodnota je uchována spolu s uživatelovým jménem a číslem 1 (čítač), které indikuje, že následující heslo pro ověření bude heslo P_1. Pokud se uživatel následně chce přihlásit k systému, zašle svůj login name autentizačnímu serveru, který mu odpoví zasláním čísla 1. Uživatel odpovídá heslem P_1, vypočítaným z s. Autentizační server následně vypočítá $f(P_1)$ a porovná tuto hodnotu s hodnotou uloženou ve své databázi jako P_0. Pokud jsou tyto hodnoty shodné, systém autentizuje uživatele, inkrementuje hodnotu čítače na hodnotu 2 a přepíše hodnotu P_0 hodnotou P_1. Takto se tak děje do doby, než je dosaženo hodnoty n. Po jejím dosažení se server reinicializuje novým tajným řetězcem s.

4.1.3 Uložení hesel

Problémem autentizace uživatelů pomocí hesel je jejich uložení v systému, který by měl toto ověření provést. Uložení hesel [4] v jejich čisté podobě by představovalo velice nedobrý způsob, ačkoliv z hlediska složitosti je toto použití nejjednodušší. Pokud by se takovýto způsob aplikoval, zadal by uživatel svůj login (přihlašovací jméno) a heslo a tato dvojice by se následně okamžitě porovnala na přesnou shodu v databázi uživatelských hesel. Slabina a racionální překážka použití je jasná – z nechráněné databáze uspořádaných dvojic [login, heslo] lze nejsnadněji získat vše potřebné.

Druhou možností je uložit heslo šifrované. Tento způsob se příliš v praxi nevyužívá, ale je možný. Nevyužití toho přístupu pramení z faktu, že není nikdy opodstatněný důvod (výjimky viz dále v práci) heslo dešifrovat do původní podoby, čímž je tento způsob zabezpečení vhodnější pro tajné zprávy, které je někdy někde nutné rekonstruovat [13].

Další z možností, jak ukládat heslo je nikoliv v jejich otevřené či šifrované podobě, ale v podobě odpovídajícího otisku (hashe). Pro získání otisku hesla se využívá rozličných hash funkcí. Při vytváření uživatelského účtu se tak z hesla hash funkcí vypočítá odpovídající otisk a v databázi se nakonec uloží dvojice [login, otisk]. Krok k vyšší bezpečnosti je v tomto případě zřejmý při kompromitaci databáze. Útočník získá uživatelská jména, ale z odpovídajících hash kódů jen těžko obdrží příslušná původní hesla. Při autentizaci legitimního uživatele je z jím zadaného hesla na straně klienta vypočítána hash kód, ten je předán druhé straně, která jej srovná s odpovídajícím

otiskem z dvojice [login, otisk] své databáze. Nastane-li shoda, uživatel se úspěšně autentizoval i bez zpracování hesla v jeho otevřené podobě.

V případě přenášení pouze otisku hesla útočník neuspěje ani s prostým odposloucháváním na síti, kdy by získal pouze login a nic neříkající hash. Nabízí se však možnost útoku přehráním, kdy útočník sice stále nezíská čitelnou podobu původního hesla, avšak budoucím zasláním odpovídající dvojice [login, otisk] by se sám mohl pokusit o přihlášení k cizímu účtu. V praxi se tento problém řeší různými časovými razítky, kdy jsou přihlašovací údaje doplněné dalším parametrem, který jim dává platnost pouze v rámci stanoveného časového okna.

Pro zvýšení bezpečnosti a pro případ, že by si dva uživatelé systému zvolili stejné heslo se při hashovaní využívá tzv. „salt“ - solení. Tato metoda je odolnější vůči útokům hrubou silou (útočník zkouší všechny možné kombinace přípustných znaků, z nich počítá hash a tu zasílá systému k ověření) a spočívá v tom, že se k heslu připojí náhodná posloupnost znaků (salt**)**. Vzniklý řetězec se až poté hashuje. Útočník tak nemůže použít již předem hashovaný slovník, ale musí slova ze svého slovníku spojit se saltem a teprve potom hashovat. Autoři vesměs zastávají postoj, že salt jako takový není nic tajného a může se klidně uložit do databáze jako další pole [13]. Tajné je však to, jak je salt použit (jestli je připojen na začátek, na konec, za druhý znak a podobně). Salt může být uložena i v úplně jiném souboru. Ověření hesla pak ale zabere o něco více času a je jen na konkrétní implementaci, jaký způsob je zvolen.

```
public void hashtext_Click(Object sender, EventArgs e)
{
 byte[] bsalt = new byte[6];
 new RNGCryptoServiceProvider().GetBytes(bsalt);
 string salt = Convert.ToBase64String(bsalt);
 byte[] bdata = Encoding.UTF8.GetBytes(tbVstup.Text + salt);
 byte[] bhash = new SHA256Managed().ComputeHash(bdata);
 string hash = Convert.ToBase64String(bhash);
 // tak a mame salted hash...
 tbSalt.Text = salt;
 tbHash.Text = hash;
}
```

Obrázek 10 - praktická ukázka programového kódu [13]

4.1.4 Útoky na hesla

Rozlišujeme dva typy útoků na hesla, kdy se útočník snaží rozluštit heslo. Útoky online a offline. Proti online útoku je snadnější se bránit, neboť je detekovatelný. Proti tomuto útoku existuje obrana dvojího druhu:

- Delayed response – zpožděná odpověď serveru, kdy po zadání přihlašovacího jména a hesla server počká, než odpoví. Pokud je tato prodleva nastavena např. na 1 sekundu, tato odradí útočníka otestovat mnoho hesel v přijatelném čase.
- Account locking – uzamčení účtu. Nastavujeme maximální počet zadání hesla, po kterém se systém uzamkne pro přihlášení.

Offline útok je nedetekovatelný ze strany systému a je proti němu těžší se bránit.

Metody, které útočníci používají pro oba typy útoků, jsou:

- Dictionary attack – slovníkový útok. Při tomto útoku využívá útočník znalost, případně předpoklad, že heslo je smysluplné slovo v daném jazyce. Tím eliminuje největší slabinu útoku hrubou silou, kterou je výpočetní náročnost.
- Brute-force attack – útok hrubou silou. Útočník zkouší všechny možné kombinace znaků, případně pokud zná pozice některých znaků v hesle, může nastavit testování pouze zbývajících pozic. Dále může ohraničit možnou délku hesla.
- Cryptanalysis attack – útok využívající tzv. Rainbow tables, neboli duhové tabulky.

Protokolů, které pracují s autentizací typu „něco znát“ existuje celá řada, přičemž výše byly zmíněny techniky, které souvisí s útoky na hesla převážně neoprávněnými osobami. Cílem tvůrců autentizačních protokolů je tyto útoky eliminovat a hledat slabiny v navržených protokolech. Jelikož se povětšinou jedná o matematický problém návrhu autentizačního protokolu, je v následující kapitole zmíněn praktický příklad protokolu, který není svázán s žádnou konkrétní platformou (Windows, Linux) a u něhož byly provedeny takové úpravy, aby eliminovaly jeho možnou zranitelnost.

4.2 Hölbl, Welzer, Brumen protokol a jeho vylepšení

V roce 2008 autoři M. Hölbl, T. Welzer a B. Brumen navrhli a v díle [11] publikovali protokol založený na heslech pro autentizaci uživatele a změnu uživatelova hesla v systému. Tento protokol chrání komunikaci mezi příjemcem a odesílatelem přes nezabezpečené médium, přičemž využívá jednoduchých matematických operací XOR a hashovacích funkcí a ve své podstatě využívá Diffie-Helmanův algoritmus na výměnu klíčů viz kapitola 9.1. Dále tento protokol využívá obousměrnou autentizaci, tedy že klient se autentizuje vůči serveru a naopak.

V rámci tohoto protokolu bylo následně v prosinci roku 2009 v díle [33] publikováno a identifikováno, že tento protokol je zranitelný vůči útoku typu „Stolen-verifier“, offline útoku na heslo a Denial-of-Service útoku. Zároveň byly navrženy postupy, jak tento nedostatek odstranit. Následuje souhrn poznatků, které se v rámci tohoto obecného autentizačního protokolu využívají,

a které byly zlepšeny pro zvýšení bezpečnosti tohoto protokolu (konkrétní vztahy, ze kterého vycházejí jednotlivé parametry lze najít ve zmíněném díle [33], zde je zmíněn pouze průběh komunikace).

V rámci tohoto protokolu se využívají tyto symboly:

- *C, S* – klient, server
- id_i – identifikátor uživatele *Ui*
- pw_i – heslo uživatele *Ui*
- *p* – velké prvočíslo
- *g* – primitivní element v GF(p)
- r_c – náhodné číslo klienta
- r_s – náhodné číslo serveru
- *x* – náhodné celé číslo klienta
- *y* – náhodné celé číslo serveru
- *H(*)* - jednosměrná hashovací funkce s argumenty
- *XOR* – operace exkluzivní OR
- $idpw\_dig_i = H(id_i , pw_i)$; autentikátor uživatele, který je využit v původní verzi protokolu
- $idpw\_dig_i = H(id_i , pw_i)$ XOR $id_i^K \bmod p$; autentikátor uživatele využitý ve vylepšené verzi protokolu. *K* je tajné celé číslo serveru.

4.2.1 Hölbl et al. protokol – původní verze (autentizace uživatele)

Předpokladem tohoto protokolu je, že server *S* má u sebe uložený autentikátor $idpw\_dig_i$.

Zde je přesný postup komunikace mezi *C* a *S* uplatněný v této verzi protokolu, přičemž tučně jsou vyznačeny položky, které jsou v průběhu komunikace zachyceny útočníkem pro využití při útoku.

1. C → S: {$\mathbf{id_i}$, $\mathbf{r_c}$, **p**, **g**, m_g^x}
2. S → C: {m_g^y, $\mathbf{ch_1}$, $\mathbf{ch_2}$}
3. C → S: { $\mathbf{id_i}$, $\mathbf{r'_s}$} (zde nejprve ověřuje C autentizaci S, je-li úspěšná, až poté zasílá svou zprávu[2])
4. S → C: {**sat**} (zde ověřuje S zaslané r'_s, je-li shodné s r_s, C se úspěšně autentizoval vůči S.)

[2] Porovnává otisk hashovací funkce *H*, který je skryt v parametru ch_2

5. C vypočítá sat´ a porovná se sat. Jsou-li shodné, autentizační token je platný.

Klíč, na kterém se *S* a *C* dohodnou má poté podobu $key = sat = H(g^{xy}, idpw\_dig_i, r_c, r_s)$

4.2.2 Hölbl et al. protokol – původní verze (změna hesla uživatele)

Uživatel *Ui* chce změnit své heslo pw_i na nové new_pw_i. Uživatel se úspěšně autentizoval na základě protokolu kapitoly 4.2.1. To znamená, že *C* i *S* znají z klíče *key* hodnoty g^{xy}, r_c a r_s. Přesný postup komunikace mezi *C* a *S* následuje:

1. C → S: {$\mathbf{id_i}$, m_idpw_dig_new$_i$, mac}
2. S → C: "accept" nebo „reject"

Server *S* před odesláním odpovědi kontroluje *mac* se svou vypočtenou hodnotou *mac´*. Pokud se tyto shodují, dovoluje změnu hesla a nahradí původní autentikátor $idpw\_dig_i$ novým $idpw\_dig\_new_i$.

4.2.3 Bezpečnostní problémy původní verze

Stolen-verifier útok

Předpokládejme, že útočník *E* získal ze serveru autentikátor $idpw\_dig_i$. V takovém případě může komunikace probíhat následujícím způsobem:

1. E → S: {$\boldsymbol{id_i}$, $\boldsymbol{r'_c}$, $\boldsymbol{p}$, $\boldsymbol{g}$, $m\_g^{x'}$}
2. S → E: {m_g^y, $\boldsymbol{ch'_1}$, $\boldsymbol{ch'_2}$}
3. E → S: { $\boldsymbol{id_i}$, $\boldsymbol{r'_s}$}

Poté, co server obdrží poslední zprávu a ověří, že r'_s je shodné s r_s, úspěšně autentizuje uživatele a klíč bude mít tedy podobu $key = H(g^{x'y}, idpw\_dig_i, r'_c, r_s)$. Jinými slovy to znamená, že útočník je schopen se autentizovat jako uživatel *Ui* a protokol je tak zranitelný vůči útoku typu Stolen-verifier.

Útok na heslo – offline

Útočník *E* zachytil komunikaci probíhající mezi *C* a *S*. Tedy zná r_c , r'_s, ch_1, ch_2 a *sat* (z předcházejícího vytučněné). *Sat* se skládá z r_c, které útočník zná, dále z r_s (předpokládáme, že uživatel *Ui* se úspěšně autentizoval a tedy můžeme psát, že $r'_s = r_s$, dále z g^{xy} (toto nezná) a $idpw\_dig_i$, ze kterého zná id_i.

Útočník tedy musí zjistit jediné, a to g^{xy}. Následně může zkoušet svá hesla pw'_i a dosazovat je do vztahu $sat' = H(g^{xy}, H(id_i , pw'_i), r_c, r_s)$. Pokud zjistí, že $sat' = sat$, zjistil správné heslo uživatele *Ui*.

Výpočet g^{xy} lze realizovat z hodnot ch_1, ch_2 a r'_s, neboť platí, že:

$$ch_1 \; XOR \; ch_2 = r_s \; XOR \; g^{xy}$$

$$g^{xy} = r_s \; XOR \; g^{xy} \; XOR \; r'_s$$

a tedy že, $\boldsymbol{g^{xy} = ch_1 \; XOR \; ch_2 \; XOR \; r'_s}$

Jak je vidět, protokol je tedy zranitelný vůči offline útoku na heslo.

Denial-of-Service útok

Tento typ útoku je využit při změně hesla uživatele, kdy se *S* nechová zamýšleným způsobem, jinak řečeno, povolí uživateli *E*, který se úspěšně do systému autentizoval, změnit heslo uživatele U_A, jehož identifikátor id_A zná. Uživatel U_A tak nebude poté schopen se do systému přihlásit svým heslem pw_A.

Platí stejné podmínky, jako při klasické změně hesla za pomoci tohoto protokolu, tedy, že *E* i *S* znají g^{xy}, r_c a r_s. Průběh komunikace bude vypadat následovně:

1. E → S: $\{id_A, m\_idpw\_dig\_new_A, mac\}$
2. S → E: "*accept*" nebo „*reject*"

Server vypočítá *mac´* a porovná ji se zaslanou *mac*. Jelikož ale tento parametr neověřuje identitu uživatele (kdo jej vytvořil), server úspěšně dovolí nahradit autentikátor $idpw\_dig_A$ novým autentikátorem $idpw\_dig\_new_A$. Protokol je tak zranitelný proti útoku typu Denial-of-Service.

4.2.4 Hölbl et al. protokol – vylepšená verze (autentizace uživatele)

Vylepšená verze protokolu počítá s tím, že autentikátor $idpw\_dig_i$ je taktéž uložen na serveru jako v základní verzi a je vypočítán dle vztahu $idpw\_dig_i = H(id_i , pw_i) \; XOR \; id_i^K \; mod \; p$, kde *K* je tajný klíč serveru. Následující komunikace probíhá tímto způsobem[3]:

1. C → S: $\{id_i , ch_1, m\_g^x\}$
2. S → C: $\{m\_g^y, MAC_1, ch_2\}$
3. C → S: $\{ MAC_2\}$ (zde nejprve ověřuje C autentizaci S z MAC_1, je-li úspěšná, až poté zasílá svou zprávu)
4. Server ověří *C* ze zaslané MAC_2, jeli totožná s vypočtenou, *C* je autentizován.

Dohodnutý klíč má poté podobu $key = H(g^{xy}, hpw_i, r_c, r_s)$, kde $hpw_i = H(id_i, pw_i)$.

---

[3] Výpočty parametrů ch_1 a ch_2 jsou různé od původních!

4.2.5 Hölbl et al. protokol – vylepšená verze (změna hesla uživatele)

Uživatel *Ui* chce změnit své heslo pw_i na nové new_pw_i. Uživatel se úspěšně autentizoval na základě protokolu kapitoly 4.2.4. To znamená, že *C* i *S* znají z klíče *key* hodnoty g^{xy}, r_c a r_s. Přesný postup komunikace mezi *C* a *S* následuje:

1. C → S: {id_i, m_ hpw_{new}, mac}
2. S → C: "accept" nebo „reject"

Server *S* před odesláním odpovědi kontroluje *mac* se svou vypočtenou hodnotou *mac´*. Pokud se tyto shodují, dovoluje změnu hesla a nahradí původní autentikátor $idpw\_dig_i$ novým $idpw\_dig\_new_i$.

Na rozdíl od parametru mac v původní verzi protokolu, tento obsahuje $H(id_i, pw_i)$, což je důležité z hlediska prokázání identity při změně hesla.

Stolen-verifier útok

Předpokládejme, že útočník *E* získal ze serveru autentikátor $idpw\_dig_i$. Neboť při odesílání první zprávy směrem k serveru potřebuje znát hpw_i, je pro něj nutné tento parametr spočítat. Z definice $idpw\_dig_i = hpw_i \; XOR \; id_i^K \; mod \; p$ je toto nemožné bez znalosti klíče *K*, který je chráněn problémem výpočtu diskrétního logaritmu. Proto je tato verze protokolu odolná vůči útoku typu Stolen-verifier.

Útok na heslo – offline

Předpokládejme, že útočník *E* zachytil uživatelovi zprávy a tedy zná ch_1, m_g^x a MAC_2. Jelikož MAC_2 obsahuje, kromě jiného i g^{xy} a vzhledem k tomu, že útočník toto nemůže vypočítat, nemůže si ověřit, že jeho zkoušené heslo pw'_i je správné. Tudíž tato verze protokolu je odolná vůči útoku typu Stolen-verifier.

Denial-of-Service útok

Předpokládejme, že útočník *E* chce změnit heslo jinému uživateli. Toto je nemožné z důvodu toho, že nový parametr $m\_ hpw_{new}$, který je zasílán při změně hesla, je vypočten jako

$$m\_ hpw_{new} = H(g^{xy}, hpw_i, r_c, r_s) \; XOR \; hpw_{new}$$

tudíž bez znalosti původního hesla, které je obsaženo v parametru hpw_i není pro něj možné toto uskutečnit.

4.2.6 Zhodnocení

Z výše uvedeného je vidět, že autentizační protokoly jsou neustále sledovány a vyvíjeny směrem k větší bezpečnosti. Chybné je zcela jistě tvrzení, že původní verze tohoto protokolu je

bezpečná, neboť jak bylo ukázáno, ta je zranitelná vůči některým typům útoku. Vylepšení, která byla navržena, těmto útokům odolávají.

4.3 Tokeny – něco mít

Při autentizaci za pomoci tokenu se využívají autentizační předměty, což může být čipová karta, USB token apod. USB tokeny pracují na stejném principu jako procesorové čipové karty, ovšem jejich výhodou je fakt, že ke své činnosti nepotřebují žádnou speciální čtečku, neboť se připojují přímo k USB portu počítače. Pro uživatele toto znamená jednodušší používání a snadnou instalaci.

Ve všeobecnosti výhoda řešení autentizace pomocí tokenů spočívá v tom, že si uživatel nemusí pamatovat přihlašovací údaje, neboť ty jsou uloženy na tokenu. V případě zcizení tohoto tokenu je ale systém zranitelný, neboť neoprávněný uživatel vystupuje vůči systému jako legitimní. Primárním požadavkem [12] na hardwareové autentizační předměty (tokeny) je zajištění vyšší bezpečnosti oproti klasickému přihlašování jménem a heslem, což vyúsťuje k využití tzv. dvoufaktorové autentizace. Ta stupeň dosažené bezpečnosti nesporně zvyšuje.

Prvním faktorem je fyzické vlastnění tokenu. Druhým faktorem je znalost PINu chránícího token před zneužitím. Token má zpravidla nastaven maximální počet neúspěšně zadaných PINů, pak se zablokuje. Znalost správného PINu je nezbytná pro aktivaci funkce tokenu. Na druhou stranu může útočník PIN odpozorovat, ale pokud nevlastní i token, není tato informace pro něj relevantní. Pro hodnocení bezpečnosti tokenu je podstatné, jaký šifrovací algoritmus je v tokenu implementován a jaká je maximální délka šifrovacího klíče.

4.4 Biometrika – něčím být

Biometrika využívá jedinečných tělesných znaků pro identifikaci osoby. Výhodou tohoto typu autentizace je, že není nutné pamatovat si několikamístné heslo či nosit sebou token. Hlavní výhoda tohoto řešení spočívá v tom, že biometrické znaky jsou během života neměnné [28].

Podstatou biometrické autentizace je tak automatizované snímání biometrických charakteristik a jejich následné porovnání s údaji, které byly sejmuty a uloženy již dříve. Využitelné se v tomto odvětví jeví vlastnosti (biologické či behaviorální), které musí splňovat tyto podmínky [28]:

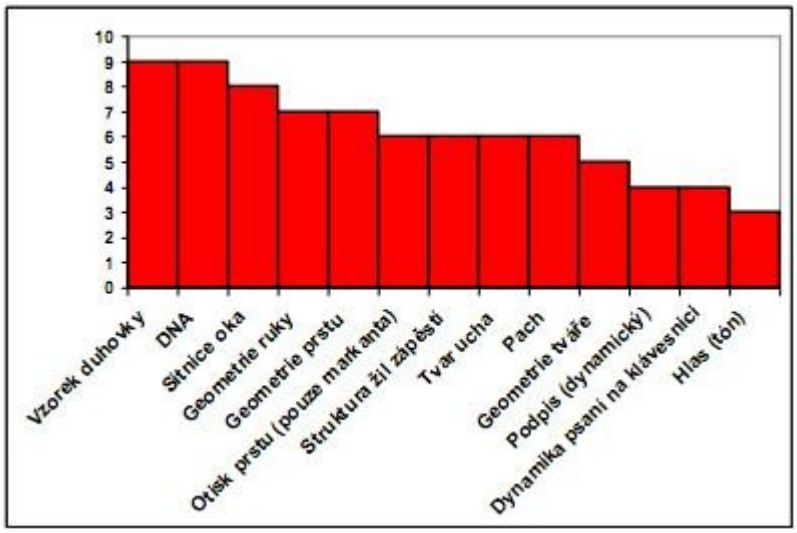
- jedinečnost – vlastnost musí být co možná nejvíce výjimečná tzn. shodná vlastnost se nesmí objevit u dvou lidí zároveň
- univerzálnost – vlastnost musí být měřitelná u co možná největší množiny lidí
- trvalost – vlastnost se s časem nemění
- měřitelnost – vlastnost se dá změřit

- uživatelská příjemnost – vyjadřuje snadnost a pohodlnost pro uživatele při měření

V tomto kontextu požadovaných vlastností na biometrické znaky jsou níže uvedeny nejvíce rozšířené znaky, které se využívají pro identifikační účely:

- otisk prstu (struktura papilárních linií)
- dynamika podpisu (rozdíl v tlaku a rychlosti psaní)
- geometrie tváře (vzdálenosti specifických částí)
- duhovka oka (obrazový vzorec duhovky)
- sítnice oka (struktura žil na očním pozadí)
- geometrie ruky (rozměry dlaně, prstů)
- struktura žil na zápěstí
- tvar ucha (rozměry viditelné části)
- hlas (tón a zabarvení)
- DNA
- pach
- psaní na klávesnici (rytmus úderů)

Z hlediska stálosti biometrických vlastností v čase lze sestavit následující graf, kde hodnota 10 znamená nejvyšší stálost.



Obrázek 11 - Stálost biometrických vlastností v čase [28]

4.4.1 Měření spolehlivosti biometrických systémů

Abychom mohli rozhodnout, který systém z hlediska chybovosti je nejspolehlivější, je třeba toto změřit definovaným koeficientem [28]. Těchto koeficientů existuje celá řada. Vždy záleží, jaké údaje od systému požadujeme. Mezi dva nejdůležitější patří

1. koeficient chybného přijetí (False Acceptance Rate – zkráceně FAR),
2. koeficient chybného odmítnutí (False Rejection Rate – zkráceně FRR).

V obou případech se jedná se o poměrové koeficienty a jsou udávány v procentech.

Ad 1)

Koeficient FAR vyjadřuje podmíněnou pravděpodobnost, že pokud se bude autentizovat neoprávněný narušitel, dojde k chybě přijetí. Jde tedy o jakousi míru bezpečnosti. Čím vyšší je tento koeficient, tím méně spolehlivý je systém při rozpoznávání uživatelů, neboť nedokáže rozhodnout, že do systému se přihlašuje neoprávněná osoba. Z tohoto se dá usuzovat na špatný návrh takovéhoto systému. Matematicky je vyjádřen následujícím vztahem:

$$FAR = \frac{N_{\text{fa}}}{N_{\text{na}}} * 100\ [\%]$$

kde N_{fa} je počet chybných přijetí a N_{na} je počet všech pokusů neoprávněných osob o autentizaci.

Tato chyba se označuje jako chyba 2. druhu.

Ad 2)

Koeficient FRR udává podmíněnou pravděpodobnost, že pokud bude autentizován oprávněný uživatel, dojde k chybě odmítnutí. Z hlediska bezpečnosti nepředstavuje tento koeficient nějaké riziko, spíše se jedná o uživatelský komfort při přihlašování. Uživatel může být nucen systémem se znovu autentizovat, neboť prvotní pokus dopadl neúspěšně, čili toto může vést k jeho nespokojenosti. Matematicky je vyjádřen následujícím vztahem:

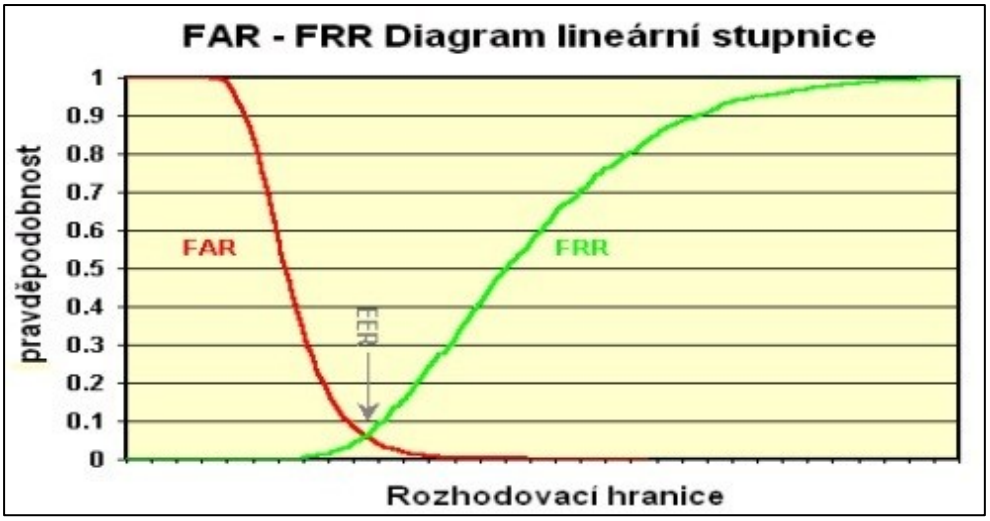
$$FRR = \frac{N_{\text{fr}}}{N_{\text{eia}}} * 100\ [\%]$$

kde N_{fr} je počet chybných odmítnutí a N_{eia} je počet všech pokusů oprávněných uživatelů o autentizaci.

Tato chyba se označuje jako chyba 1. druhu.

Ukazatelem, který bere obě výše zmíněné podmíněné pravděpodobnosti v potaz a je rozhodujícím při vlastním nastavování biometrického systému je koeficient EER (Equal Error Rate), který stanový, při jaké pravděpodobnosti a nastavení systému nastane jev FAR a FRR současně, jinými slovy kdy se FAR = FRR. Tento koeficient tedy vypovídá o tom, zda bezpečnost

a uživatelská přívětivost biometrického systému jsou v rovnováze. Pokud budeme posouvat hranicí rozhodování, vždy se tedy systém stane bezpečnějším na úkor přívětivosti a opačně.



Obrázek 12 - Distribuční pravděpodobnostní funkce FAR – FRR [28]

5 Autentizace ve Windows

V této kapitole se budu zabývat autentizačními protokoly, které jsou využívány systémy Microsoft Windows[4] pro autentizaci uživatelů za pomoci hesel.

Existují dva různé typy uživatelů [14]: lokální (locals) a doménoví (domains). Lokální uživatelé jsou definováni v lokální databázi Security Accounts Manager (SAM) daného počítače a každý počítač má svou vlastní SAM, která obsahuje všechny uživatele daného počítače. Doménoví uživatelé resp. uživatelé, kteří mají doménový účet, jsou definování v tzv. Doménovém řadiči – Domain Controller (DC) a mohou využít kterýkoliv počítač v příslušné doméně.

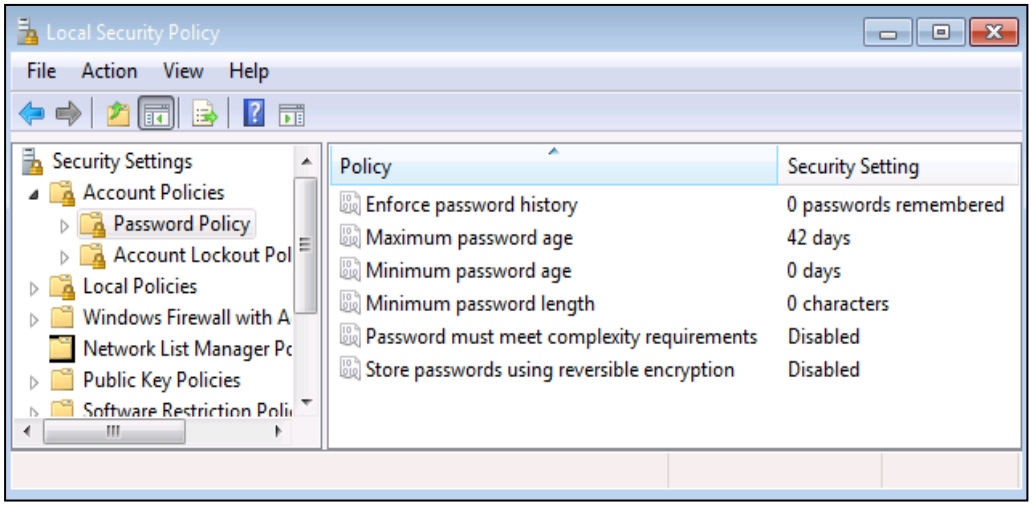
5.1 Hesla – bezpečnost v systému Windows

V předcházejících kapitolách byly zmíněny způsoby, kterými lze uložit heslo pro pozdější ověření. Ať už se jedná o způsob uložení v otevřené formě, ve formě zašifrované, hashované, případně doplněné „solí“. V následující části budou zmíněny techniky, se kterými pracují systémy Windows, přičemž nejprve budou zmíněny zásady jejich bezpečnostních politik.

5.1.1 Politika hesel

Z hlediska bezpečnosti hesla je nutné, aby uživatel byl upozorňován na situaci, kdy jeho zvolené heslo neodpovídá politice, která byla v systému přijata. Politikou hesel chápeme to, jak dlouhé heslo musí být, z jakých znaků a symbolů se musí skládat, jaká je platnost tohoto hesla atd. V systémech Windows je tato politika nastavována za pomoci služby *secpol.msc*, kterou lze vyvolat příkazem Start-Spustit-secpol.msc. Část, která se zabývá nastavením výše uvedeného, se nazývá *Account Policy*, která je rozdělena na dvě části: politiku hesel jako takovou (*Password Policy*) a politiku uzamčení účtů (*Account Lockout Policy*). V systému Windows 7 vypadá *Password Policy* následovně:

[4] Za systémy Windows budou v této práci dále považovány tyto verze: NT 3,1; NT 3,5;, NT 3,51; NT 4.0;, 2000, XP, Server 2003, Vista a Server 2008.

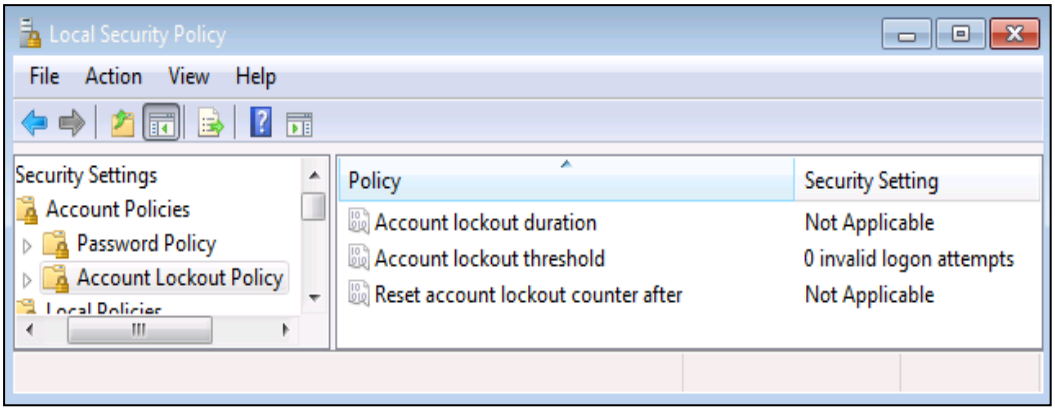


Obrázek 13 - Password Policy [zdroj: vlastní]

Jednotlivé politiky lze charakterizovat takto:

- Enforce password history – počet hesel, které si systém pamatuje a kterým brání je opět použít.
- Maximum password age – maximální doba platnosti hesla.
- Minimum password age – minimální doba platnosti hesla.
- Minimum password lenght – minimální délka hesla.
- Password must meet complexity requirements – pokud je povoleno, uživatelovo heslo nesmí obsahovat uživatelské jméno, případně jeho část a musí se skládat minimálně ze tří abeced, kdy na výběr je [A-Z]; [a-z]; [1-9] a abeceda speciálních znaků (např. ! “ : /). Délka hesla musí být větší než 6 znaků.
- Store passwords using reversible encryption – zda systém uchovává hesla v podobě zpětně dešifrovatelné (reversibly encrypted viz kapitola 5.1.4)

Pokud jde o politiku *Account Lockout Policy*, ta vypadá následovně:

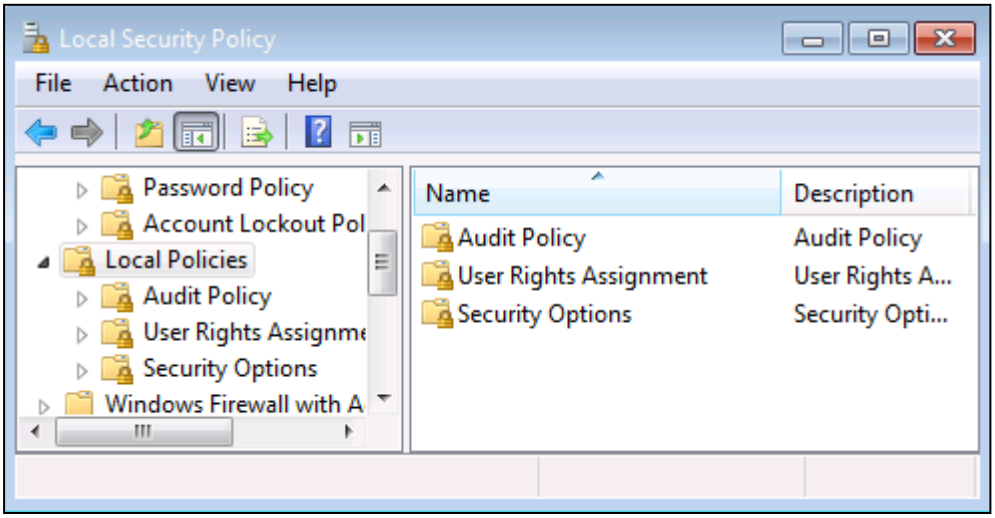


Obrázek 14 - Account Lockout Policy [zdroj: vlastní]

Jednotlivé politiky lze charakterizovat takto:

- Account lockout duration – určuje dobu v minutách, jak dlouho zůstane účet uzamčen, dokud se automaticky neodemkne.
- Account lockout threshold – určuje počet neplatných pokusů o přihlášení, po kterém se účet uzamkne do doby, než jej administrátor odemkne, případně než uplyne doba nastavená v parametru account lockout duration.
- Reset account lockout counter after – určuje počet minut, které uplynou po neplatném přihlášení, než je počitadlo chybných přihlášení opět vynulováno.

Další částí politik služby secpol.msc je část *Local Policy* zobrazená níže:



Obrázek 15 - Local Policies [zdroj: vlastní]

Tato politika obsahuje mnoho nastavení týkajících se auditování a logování činností, které se týkají systému (např. logování přihlašování, změn bezpečnostních politik apod.). Z hlediska autentizace jsou zde ovšem důležité položky v sekci *Security Options*, kde lze například v klíči *Network security: LAN Manager authentication level* nastavit, který protokol (LM, NTLM, NTLMv2) bude využíván v procesu síťového přihlašování. Co se týče protokolu Kerberos, lze vyhledat klíč *Network security: Configure encryption types allowed for Kerberos*, kde lze stanovit, které šifrování bude tímto protokolem využíváno[5].

5.1.2 Techniky ukládání hesel - LM Hash

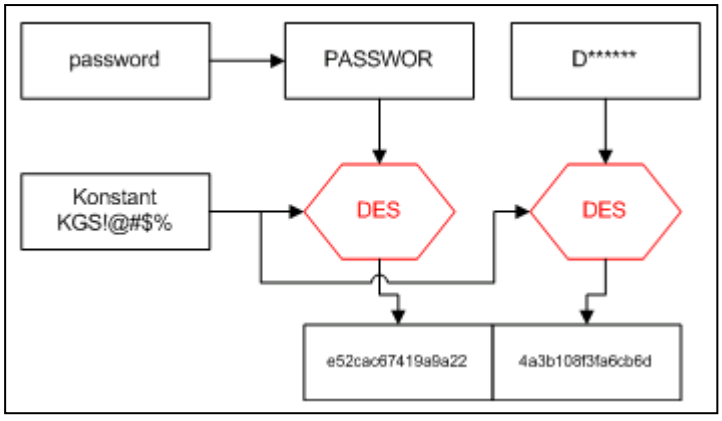
LM Hash nelze téměř vůbec považovat za hash tak, jak byla dříve definována, ačkoliv je tu jistá podobnost. Je to jednosměrná funkce, která je všeobecně známá pod označením LM OWF

[5] Dostupné u Windows 7 a Windows Server 2008 R2

(LanManager One-Way Function) [20]. Ve Windows Vista, případně Windows Server 2008 LM Hash není ve výchozím nastavení vůbec ukládána, ani není použita během procesu autentizace, ale nižší verze Windows LM Hash používají, ukládají a přenášejí[6]. Proto je znalost toho, jak pracuje velice důležitá. Z ní lze dovodit slabinu autentizačních protokolů založených právě na této LM Hashi.

LM hash je vytvářena následovně: [2]

- Uživatel vytvoří nové heslo. Toto heslo je okamžitě převedeno na velká písmena, tzn. hesla uložena za pomoci LM Hashe jsou necitlivá na velikost písmen.
- Jakmile je heslo převedeno na velká písmena, je doplněno do 14 znaků. Pokud je heslo delší, teoreticky může být zkráceno, ale v praxi se proces vytváření hesla přeruší. Z tohoto důvodu se lze setkat s varováním systému, že pro zachování zpětné kompatibility se doporučuje volit hesla kratší 14-ti znaků.
- Následně je heslo rozděleno na dvě 7mi místné části, neboť budou sloužit jako vstup do šifrovacího algoritmu DES (více o tomto algoritmu v kapitole 9.2).
- Posledním krokem algoritmu je spojení obou výstupů ze šifry DES a uložení tohoto výsledku jako LM Hash. Tato hash je následně uložena v SAM (pokud se jedná o lokální účet na samostatném počítači), případně je uložena v atributu DBCS-Pwd, jedná-li se o uživatele v Active Directory.



Obrázek 16 - LM hash [31]

Výše popsaný mechanismus zároveň ukazuje slabinu celého systému. Útočník totiž velice lehce odhadne délku hesla daného uživatele pouhým pohledem na LM Hash. Je-li totiž druhá polovina

[6] Vista i Server 2008 mohou být nakonfigurovány tak, aby LM Hash používali, ačkoliv se to nedoporučuje právě z důvodu nižší bezpečnosti

LM hashe ve tvaru „AAD3B435B51404EE“, je druhá polovina uživatelova hesla prázdná. Jsou-li obě strany LM Hashe v tomto tvaru, je celé uživatelovo heslo prázdné!

Další, co dělá tento algoritmus zranitelným je způsob, jakým lze dojít ke správnému heslu útokem hrubou silou, tedy vyzkoušením všech možných kombinací. V tomto případě jde útočníkovi na pomoc několik věcí:

- útočníkovi je známo, že musí prohledat kombinace pouze velkých písmen, což mu značně redukuje abecedu.
- útočníkovi stačí rozdělit celou LM Hash na dvě poloviny a testovat je samostatně (pochopitelně pouze v tom případě, že i druhá polovina má jinou hash, než je uvedena výše), což pro něj znamená vyzkoušení kombinací od 1 do 7 znaků.

Z tohoto je patrné doporučení, proč nevyužívat tento způsob ukládání hesla v systémech Windows.

Tabulka 1 ukazuje bezpečnostní rizika této hashe na praktickém příkladě. Jako heslo byl zvolen textový řetězec „MojeHeslo“, přičemž na postupném přidávání jednotlivých znaků byly demonstrovány výše uvedená fakta.

Tabulka 1 - Praktický příklad LM Hashe [zdroj: vlastní]

Počet znaků hesla	Heslo	LM hash
1	M	1486235A2333E4D2**AAD3B435B51404EE**
2	Mo	A33D276828F74E4B**AAD3B435B51404EE**
3	Moj	85A63484CFA77B24**AAD3B435B51404EE**
4	Moje	244D92FF490A4934**AAD3B435B51404EE**
5	MojeH	6E89BD51731AACC7**AAD3B435B51404EE**
6	MojeHe	6FD8B931AB95DE40**AAD3B435B51404EE**
7	MojeHes	**923EED98CF0B2862AAD3B435B51404EE**
8	MojeHesl	**923EED98CF0B2862**F500944B53168930
9	MojeHeslo	**923EED98CF0B2862**E917F8D6FA472D2C

5.1.3 NT Hash

Poprvé se tento způsob uložení hesla objevil v roce 1993 při příchodu systému Windows NT[14]. Jeho mechanismu je mnohem jednodušší, než mechanismus tvorby LM Hashe. Skládá se pouze ze dvou kroků:

- Uživatel vytvoří nové heslo

- Z tohoto hesla je vytvořen otisk využívající hashovací funkci MD4[7] a v této podobě je heslo uloženo.

Uložení NT hashe se opět uskutečňuje v lokální SAM, případně v atributu Unicode-PWD, jedná- li se o uživatele Active Directory.

Z výše popsaných algoritmů je zřejmé, že v systémech Windows se nevyužívá tzv. „solení“. Má to svůj důvod, kterým je to, že databáze hesel není nikdy čitelná pro ostatní uživatele. K jejímu přístupu je v prvé řadě potřeba být administrátor, čili je zde předpoklad, že útočník by musel získat plný přístup k počítači, případně doméně.

Tabulka níže ukazuje srovnání obou hashí, které následně najdou své uplatnění v autentizačních protokolech LM a NTLM.

Tabulka 2 - Komparace LM a NT hash [zdroj: vlastní]

	LM hash	NT hash
Délka hesla (max.)	14 znaků	256 znaků
Dělení hesla	2 x 7 znaků	Ne
Šifrovací algoritmus	DES	MD4
Abeceda	Necitlivá na velikost písmen	Citlivá na velikost písmen
Délka klíče	56 bit	128 bit

5.1.4 Uložení hesel

Pokud uživatel pracuje s počítačem, který je součástí prostředí Active Directory a tento počítač přenáší, lze si všimnout toho, že i přesto může využít svůj doménový účet, aniž by byl do domény přihlášen. Je tedy jasné, že i doménové heslo je na počítači lokálně uloženo.

Je uloženo ve formě hashe doménového hesla a tuto lze využít při lokálním přihlašování. V operačních systémech uvedených na trh před systémem Windows Vista byla tato hash vytvářena následujícím způsobem[14]:

- z uživatelova hesla se vypočítá MD4 hash
- k této hashi je přidána „sůl“ v podobě uživatelova přihlašovacího jména
- z tohoto celku je opětovně vypočítána MD4 hash

Ačkoliv se jedná o vcelku bezpečnou metodu (hash hashe s přidáním „soli“) má svou slabinu v případě, že heslo není příliš silné. Z tohoto důvodu systémy Windows Vista a pozdější tento

[7] Příslušné RFC: http://www.faqs.org/rfcs/rfc1320.html

algoritmus modifikovaly a přidaly na jeho konec operace od společnosti RSA, The Security Division of EMC definované v dokumentu PKCS #5[8] díky čemuž se zvedla bezpečnost celého systému.

Dalším místem, kde jsou hesla ukládána, je paměť počítače, která je vyhrazena pouze pro operační systém, případně pro proces, který má oprávnění operačního systému. V tomto případě se jedná o hash (NT Hash, nebo LM Hash, pokud je systém nakonfigurován ukládat i tuto „slabou" hash), která je uložena do paměti v momentě, kdy se uživatel interaktivně přihlašuje, případně využívá terminálové služby. Následně pokud se uživatel pokusí přistoupit k síťovému zdroji, který vyžaduje autentizaci, operační systém využije tuto hash, aby pomocí ní uživatele autentizoval. Jakmile se uživatel odhlásí, případně uzamkne stanici, místo v paměti se automaticky uvolní.

Posledním způsobem uložení hesla v systémech Windows je uložení hesla v podobě zašifrování, které se dá převést zpět na podobu původního hesla (tzv. *reversibly encrypted* – zpětně dešifrovatelné), což je v rozporu s filozofií hashí a s tím, že by hesla neměla být ukládána v čisté podobě. Toto uložení hesel je ve výchozím nastavení vypnuto a z hlediska využitelnosti se stává potřebné pouze ve dvou případech [14]:

- uživatel se přihlašuje pomocí starších autentizačních protokolů pro vzdálený přístup, jako je CHAP[9], případně Digest protokol, nebo
- pokud chceme udělat zevrubnou analýzu hesel po jejich nastavení. Jinými slovy to znamená zjistit, jaká hesla uživatelé používají, zda jsou zranitelná apod.

Kromě výše zmíněných důvodů není proč hesla uchovávat v takovéto podobě.

5.2 Autentizační protokoly

V této kapitole budou zmíněny základní autentizační protokoly systémů Windows. Pokud se uživatel přihlašuje interaktivně k lokálnímu účtu, následuje tento postup[14]:

1. Uživatel stiskne SAS sekvenci (Secure Attention Sequence – Ctrl+Alt+Del) k vyvolání přihlašovacího dialogu. To způsobí, že LSASS (Local Security Authority SubSystem) otevře novou session a spustí v ní proces WinLogon. Ten následně volá LogonUI.
2. Uživatel vloží své jméno a heslo

[8] K nalezení zde: http://www.rsa.com/rsalabs/node.asp?id=2127

[9] Více zde: http://tools.ietf.org/html/rfc1994

3. WinLogon proces toto heslo zahashuje do podoby NT hashe, vyhledá uživatelovo jméno v lokální databázi SAM a porovná NT hash s tou, kterou má v SAM uloženou. Pokud jsou tyto hashe stejné, uživatel se úspěšně autentizoval.
4. Pokud jsou na dané stanici instalovány i další autentizační balíčky, přihlašovací údaje jsou jim následně zaslány k jejich zpracování. Pokud ne, uživateli se rovnou zobrazí jeho prostředí.

Jedná se o přímočarý proces přihlášení, neboť od vložení dat uživatele jsou tato po celou dobu zpracovávána zabezpečeným kanálem.

V procesu přihlášení je vidět i místo, kde jsou data uložena. Jedná se o SAM, což je vyhrazená část registru v klíči HKEY_LOCAL_MACHINE\SAM. Na pevném disku je tento soubor uložen v adresáři %SystemRoot%\system32\config\SAM.

V případě vzdáleného přihlašování pomocí počítačové sítě je třeba dát pozor, jakým způsobem je zabezpečen přenos autentizačních údajů mezi klientem a serverem.

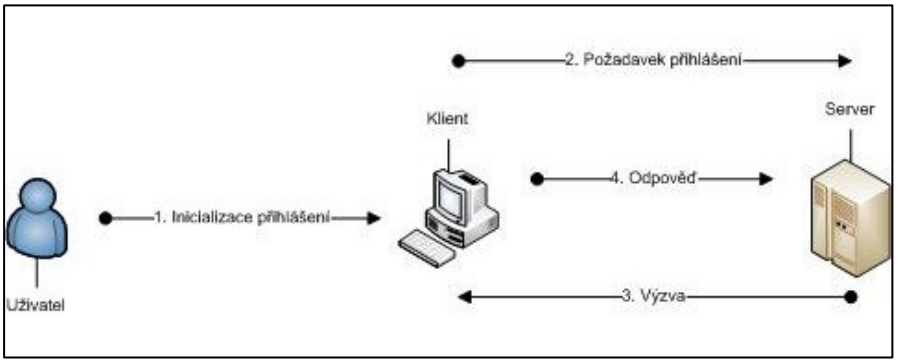
5.2.1 Password Authentication Protocol (PAP)

PAP je nejjednodušší formou autentizace v systémech Windows. Při tomto způsobu se přihlašovací údaje přenášejí v síti v nešifrované podobě, případně zakódované pomocí Base64[10].

Tento typ autentizace je běžný ve starších síťových protokolech jakými jsou FTP, POP, IMAP [14].

5.2.2 Protokoly typu výzva – odpověď

Základním principem protokolů založených na tomto schématu je, že komunikaci mezi serverem a klientem probíhá na základě následujícího schématu:



Obrázek 17 - Protokol typu Výzva-Odpověď [zdroj: vlastní]

[10] Popis kódování lze nalézt na http://cs.wikipedia.org/wiki/Base64

1. Klient zašle požadavek na server s žádostí o přihlášení
2. Server vytvoří „výzvu“ (často se jedná o náhodné číslo) a zašle ji klientovi
3. Uživatel následně zkombinuje své přihlašovací údaje s výzvou danou kryptografickou metodou
4. Tuto kombinaci odešle na server jako „odpověď“

Cílem, proč implementovat autentizaci tímto způsobem je požadavek, aby heslo nemuselo být přenášeno v otevřené podobě pomocí nezabezpečeného kanálu. Protokoly založené na tomto principu, které používají systémy Windows, jsou zmíněny v další části práce a vycházejí z [14].

Digest protocol

Protokol Digest je navržen tak, aby nahradil autentizaci za pomoci protokolu PAP. Využívá se v IIS (řešení společnosti Microsoft – sada internetových služeb) v souladu s RFC2617[11]. Autentizace na základě tohoto protokolu se skládá z těchto kroků [14]:

1. Server vygeneruje náhodné číslo a zašle jej klientovi.
2. Klient vypočítá MD5 hash z uživatelského jména, názvu domény a hesla
3. Klient vypočítá MD5 hash metody (GET, POST) a digest URI (identifikátor toho, co požaduje např. /text/slovnik.html)
4. Klient vypočítá MD5 hash z výsledku kroku 2, náhodného čísla zaslaného v kroku 1, čísla autentizační žádosti, kódu qop (udává kvalitu zabezpečení – „auth“ nebo „auth-in“), klientova náhodného čísla a z výsledku kroku 3. Toto je jako odpověď zasláno na výzvu serveru.
5. Server vypočítá stejné hodnoty a následně porovná.

Jak je vidět, důležité se odehrává v kroku 2, kde se vypočítává MD5 hash z otevřené formy hesla uživatele, nikoliv z jeho hashe. Ve výsledku to znamená, aby byl server schopen porovnat své výpočty z kroku 5 se zaslanými, musí mít k dispozici stejné údaje, jinými slovy musí mít uloženo uživatelovo heslo též v otevřené podobě. Z tohoto důvodu, pokud je tento typ autentizačního protokolu umožněn, je nutné uchovávat hesla v podobě *reversibly encrypted* viz kapitola 5.1.4.

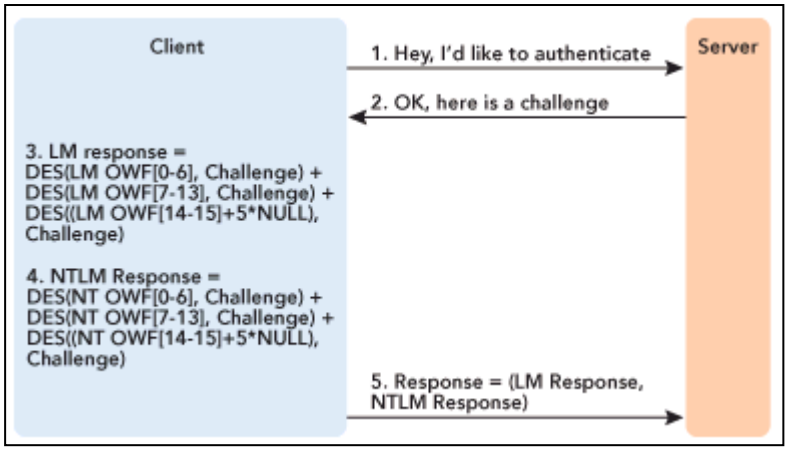
LM a NTLM

Protokoly LM a NTLM jsou si podobné. Jediným rozdílem je hash, kterou používají pro výpočet odpovědi [14]. Oba se používají při lokální autentizaci v systémech založených na jádru Windows-NT, případně, pokud není klient či server členem domény, v doménovém prostředí. Další

[11] Definice zde http://www.faqs.org/rfcs/rfc2617.html

využití najdou tyto protokoly v případech, kdy je požadovaná služba (zdroj) specifikována svou IP adresou a ne svým jménem (*host name*). Pokud se jedná o doménu Active Directory, je využíván protokol Kerberos (viz dále). Důvodem, proč jsou tyto protokoly využívány v případech, kdy je zdroj specifikovaný svou IP adresou, je fakt, že protokol Kerberos je založen na plně kvalifikovaném doménovém jméně (FQDN – Fully Qualified Domain Name) a toto nelze z IP adresy získat, neboť každý počítač může mít několik aliasů.

Veškeré systémy založené na jádře Windows-NT před příchodem Windows Server 2003 pracují na tomto principu:



Obrázek 18 - LM a NTLM schéma autentizace [15]

Z obrázku je vidět, že se zasílají obě odpovědi najednou (LM i NTLM). Od uvedení Windows Server 2003 je posílána pouze NTLM odpověď, přičemž LM odpověď je prázdná. Oba protokoly akceptují příchozí požadavky na autentizaci, což se s příchodem Windows Vista a Windows Server 2008 změnilo.

NTLMv2

Od příchodu Windows Vista a dále Windows Server 2008 jsou předcházející LM a NTLM protokoly omezeny. Protokol NTLM je podporován pro příchozí požadavky, ale pro odchozí je již používána vylepšená verze tohoto protokolu NTLMv2 [14]. Předcházející verze Windows systémů mohou být nakonfigurovány tak, aby se chovaly stejně, ale není to jejich výchozí nastavení. Jinými slovy, počítač akceptuje autentizační požadavek na bázi protokolu LM, ale ve výchozím nastavení ani Windows Vista, ani Windows Server 2008 neuchovávají LM hash. Z tohoto důvodu nemohou odpovědět již dříve definovanou LM odpovědí.

Autentizační prostředí se v nových operačních systémech Windows (od Windows NT 4.0 SP4) dá nastavit přes LMCompatibilityLevel v registru systému[12]. Zde je nutné do tohoto klíče vložit číselnou hodnotu v rozmezí 1 – 5, přičemž význam těchto hodnot ukazuje tabulka níže. Další možností je např. v systému Windows Vista spustit secpol.msc, která přehledně zobrazí nejen toto nastavení.

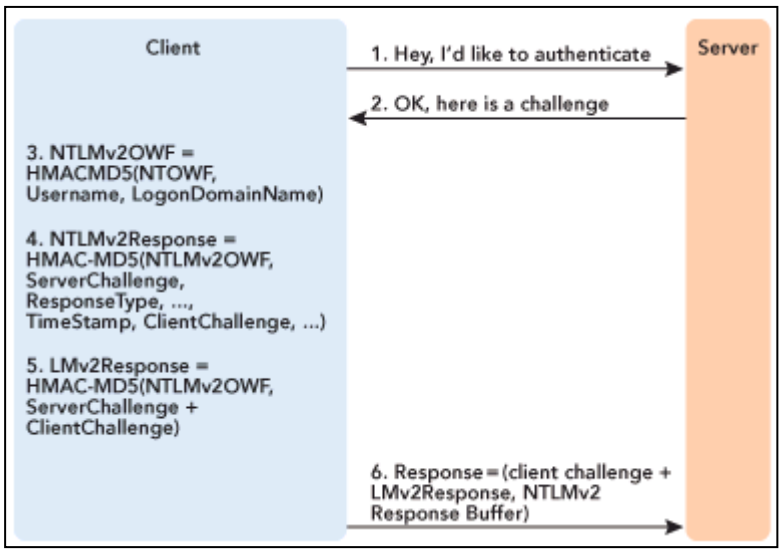
Tabulka 3 - Hodnoty klíče LMCompatibilityLevel [22]

Hodnota	Význam
0	Odesílat odpovědi LM a NTLM, nikdy nepoužívat zabezpečení relace NTLM 2. Klienti používají ověřování LM a NTLM a nikdy nepoužívají zabezpečení relace NTLM 2; řadiče domén přijímají ověřování pomocí protokolů LM, NTLM a NTLM 2.
1	Zabezpečení relace NTLM 2 se používá, pokud je vyjednáno. Klienti používají ověřování LM a NTLM a zabezpečení relace NTLM 2 používají v případě, že je server podporuje; řadiče domén přijímají ověřování pomocí protokolů LM, NTLM a NTLM 2.
2	Odesílat pouze odpovědi NTLM. Klienti používají pouze ověřování NTLM a zabezpečení relace NTLM 2 používají v případě, že je server podporuje; řadiče domén přijímají ověřování pomocí protokolů LM, NTLM a NTLM 2.
3	Odesílat pouze odpovědi NTLM 2. Klienti používají ověřování NTLM 2 a zabezpečení relace NTLM 2, pokud je server podporuje; řadiče domén přijímají ověřování pomocí protokolů LM, NTLM a NTLM 2.
4	Řadiče domén odmítají odpovědi LM. Klienti používají ověřování NTLM a zabezpečení relace NTLM 2, pokud je server podporuje; řadiče domén odmítají ověřování LM (tedy přijímají ověřování NTLM a NTLM 2).
5	Řadiče domén odmítají odpovědi LM a NTLM (přijímají pouze NTLM 2). Klienti používají ověřování NTLM 2 a zabezpečení relace NTLM 2, pokud je server podporuje; řadiče domén odmítají ověřování NTLM a LM (přijímají pouze ověřování NTLM 2).

Klientský počítač může při komunikaci se všemi servery používat pouze jeden protokol. Nelze jej nakonfigurovat tak, aby například pro připojení k serverům se systémem Windows 2000 používal NTLM 2 a pro připojení k ostatním serverům používal NTLM. Toto chování je záměrné [14].

Protokol NTLMv2 je vylepšenou verzí protokolu NTLM, který taktéž používá NT hash. Navíc ale při autentizaci a výpočtu využívá i klientovu výzvu. Zároveň vypočítává i dvě HMAC-MD5 pro vytvoření odpovědi. Dále využívá i časové razítko k bránění se proti útoku přehráním (zachycení hesla a jeho následné využití). Obrázek níže ukazuje, že v odpovědi je zahrnuta i odpověď typu LMv2, jenž je konstantní délky a je zde z důvodu zpětné kompatibility např. při autentizaci v systémech Windows 95.

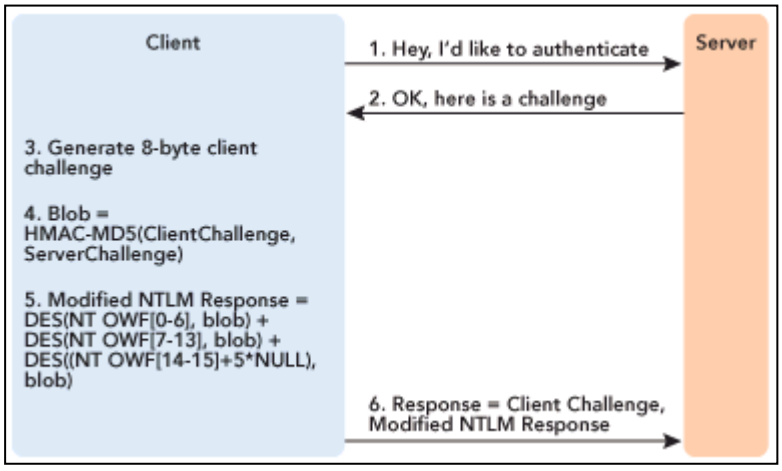
[12] HKLM\SYSTEM\CurrentControlSet\Control\Lsa



Obrázek 19 - NTLMv2 schéma autentizace [15]

NTLMv2 Session Security

NTLMv2 SessionSecurity, bývá označován někdy jako protokol NTLM2, přičemž již výše zmíněný protokol NTLMv2 by se v této souvislosti měl přejmenovat na NTLM3. Jedná se o protokol, který Microsoft přidal do své rodiny NTLM protokolů z důvodu zvýšení bezpečnosti proti útokům typu man-in-the-middle v případě, že se jedná o autentizaci vůči serverům běžícím na dřívějších verzích operačního systému Windows [15].



Obrázek 20 – Odpověď při nastavení LMCompatibilityLevel 1 nebo 2 [15]

Z obrázku výše je zřejmé, že odpověď obsahuje modifikovanou NTLM odpověď, která je vypočítána stejným způsobem jako originální NTLM odpověď pouze s tím rozdílem, že používá HMAC-MD5 funkci na výzvu klienta i serveru, zatímco originální používá pouze výzvu serveru.

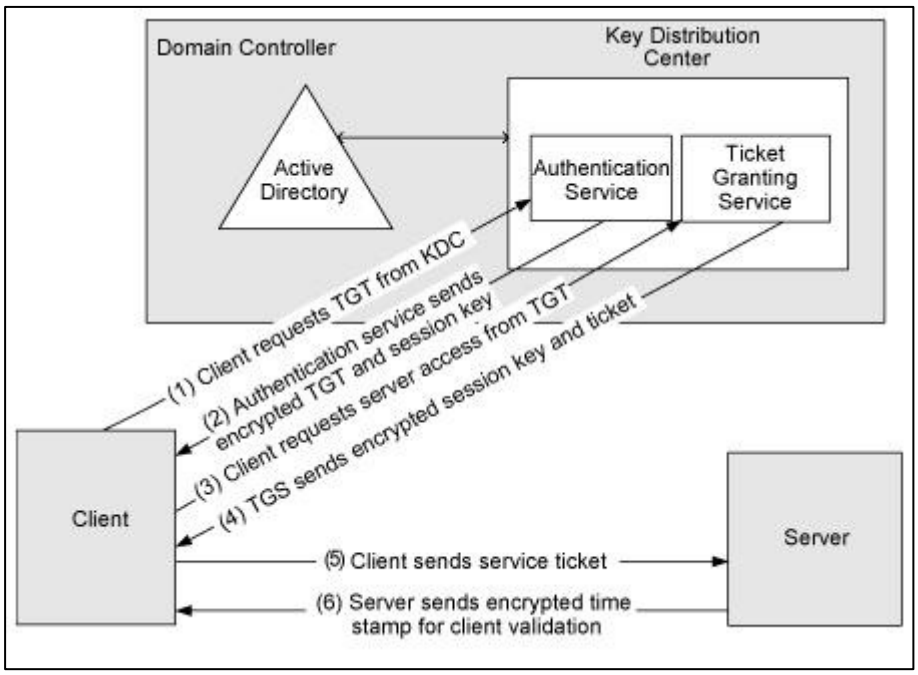
Kerberos

Protokol Kerberos je využíván v doménovém prostředí v případech, kdy jsou pro připojení využívána doménová jména (host name). Většinou je tomu tak s výjimkou případů, kdy uživatel

specificky uvede, že se požaduje připojit na danou konkrétní IP adresu. Stejně tak, jako protokol NTLM je i protokol Kerberos ve Windows implementován jako zprostředkovatel zabezpečení (SSP) a taktéž využívá pro autentizaci NT hash [14].

Protokol Kerberos zajišťuje autentizaci pro obě komunikující strany. U rodiny protokolů NTLM neměl uživatel jistotu, že server, ke kterému se přihlašuje, je skutečně tím, za koho se vydává. V případě protokolu Kerberos tomu tak je. Zároveň tento protokol předpokládá i nepřátelskou počítačovou síť tzn., že kdokoliv může zachytit provoz na síti a může tento provoz přesměrovávat, modifikovat atp.

Základní schéma přihlašování ukazuje následující obrázek:



Obrázek 21 - Kerberos schéma autentizace [21]

1. V prvním kroku [21] uživatel požaduje tzv. Ticket Granting Ticket (TGT) a tak zašle žádost (request) směrem ke Key Distribution Center (KDC). Žádost obsahuje uživatelovo jméno, informace o službě, ke které chce uživatel přistupovat za pomoci TGT a časové razítko, které je zašifrováno heslem.
2. KDC získá heslo pro daného uživatele z Active Directory a dešifruje časové razítko, které bylo zasláno v prvním kroku. Pokud je toto časové razítko platné, uživatel je ověřen. Dále KDC vytvoří logon session key a jeho kopii zašifruje uživatelovým heslem. Následně Authentication Service (AS) vytvoří TGT lístek, který obsahuje

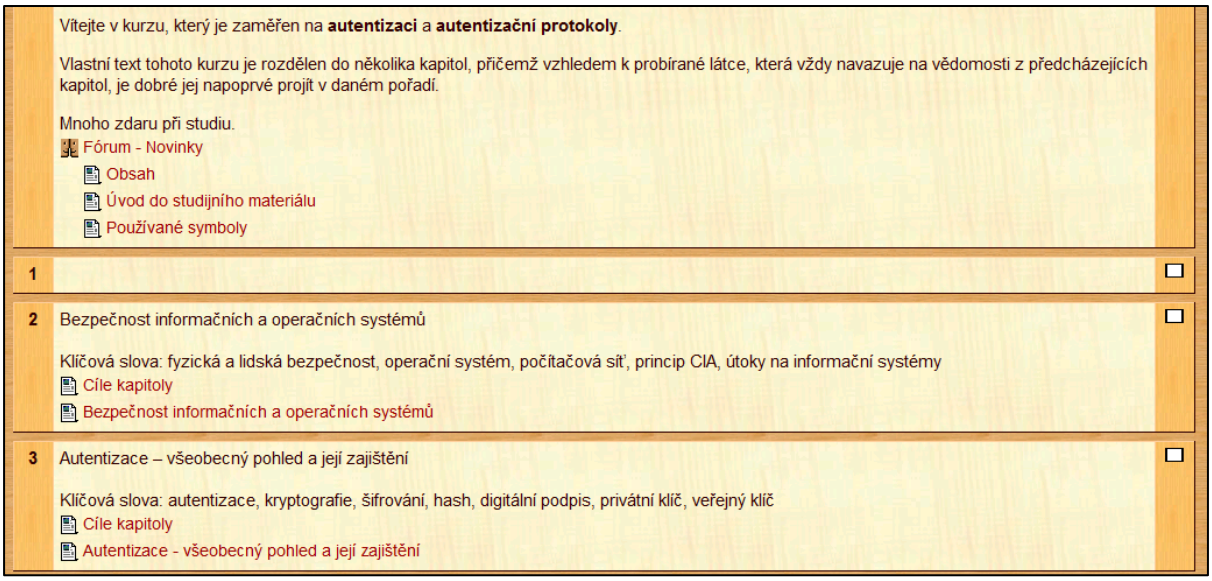
uživatelovo informace a logon session key, který nakonec AS zašifruje svým klíčem. Následně zašle zašifrovaný logon session key a zašifrovaný TGT uživateli.

3. Klient dešifruje logon session key za použití svého hesla a lokálně si jej uloží do své cache. Tam si též uloží zašifrovaný TGT lístek. Pokud bude chtít uživatel přistoupit k nějaké síťové službě, klient zašle žádost na Ticket Granting Service (TGS) s těmito informacemi: uživatelské jméno, autentikátor zprávy (většinou časové razítko) zašifrovaný uživatelovým logon session key a dále žádost obsahuje TGT a jméno služby (a serveru), ke které chce uživatel přistupovat.
4. TGS dešifruje TGT za použití svého klíče a dále pomocí logon session key dešifruje autentikátor zprávy. Pokud je tento úspěšně dešifrován a je platný, TGS získá informace o uživateli z TGT a na základě nich vytvoří service session key pro přístup k požadované službě. Jednu kopii service session key zašifruje TGS uživatelovým logon session key a dále ze service session key a uživatelových informací vytvoří service ticket, který zašifruje serverovým heslem. Následně zašle TGS uživateli zašifrovaný service session key a service ticket.
5. Pokud klient přistupuje ke službě, zasílá serveru žádost, která obsahuje: autentikátor zprávy (časové razítko), které je zašifrováno pomocí service session key a service ticket.
6. Server dešifruje service ticket. Za použití service session key server dešifruje autentikátor zprávy. Pokud je platný, zašifruje server autentikátor pomocí service session key a zašle jej zpět uživateli. Uživatel dešifruje autentikátor a pokud je shodný s originálem, služba je důvěryhodná, autentická a uživatel ji tak může využívat.

Výše popsaný mechanismus nazýváme též *Single Sign-On* princip, kdy uživatel využívá služeb sítě bez zbytečných bezpečnostních procedur, neboť se vůči serveru autentizuje pouze jednou a nadále, pokud jsou vyžadovány nějaké autentizační mechanismy, se uživatel na pozadí prokazuje TGT lístkem, který od serveru obdržel.

6 Moodle

V rámci této diplomové práce byla vytvořena učební jednotka – studijní text, který byl vytvořen v online vzdělávacím prostředí Moodle. Jelikož se jedná o materiál určený pro distanční vzdělávání, byla při jeho zpracování využita příručka tvorby distančních opor Centra distančního vzdělávání [8]. Výsledkem je tak učební text rozdělený do několika kapitol, přičemž na začátku každé z nich jsou definovány klíčová slova, cíle dané kapitoly a čas potřebný k jejímu zvládnutí. Následně je zpracován vlastní text distanční opory dle doporučení spolu s kontrolními otázkami a odkazy na literaturu. Obsahem této opory je i závěrečný test na téma autentizace a autentizační protokoly.



Obrázek 22 - Náhled výukového materiálu z prostředí Moodle [zdroj: vlastní]

Kompletní výukový materiál je k dispozici na CD-ROM, který je přílohou této diplomové práce.

7 Závěr

Tato práce si kladla za cíl představit vybrané kapitoly z tématu autentizace. Byly zmíněny základní šifrovací a autentizační algoritmy, přičemž následně byl důraz kladen na prostředí Microsoft Windows. Je faktem, že autentizační protokoly jsou vždy poplatné době. Využívajíc šifrovacích metod a s tím, jak roste výpočetní výkon (kdy v případě hashí lze ke kolizi, tedy najít jiný řetězec, který má stejný otisk jako např. uživatelovo heslo, dojít v přijatelném čase), je nutné být neustále připraven. Softwarové společnosti vytvářející operační systémy jsou si tohoto nebezpečí zcela jistě vědomi, ale i tak (např. Microsoft) nechávají uživateli možnost ukládat svá hesla ve zpětně dešifrovatelné podobě, což může být obecně vzato jako velice riziková záležitost, vždyť pokud se útočník dostane k takovýmto údajům a tato hesla zpětně dešifruje, má otevřený přístup do systému. Ve všeobecnosti můžeme říct, že se ani nemusí jednat o hesla zpětně dešifrovatelná, ale i o hashe, které jsou tím sdíleným tajemstvím v procesu autentizace. Ovšem dostat se k těmto údajům je velice obtížné. Zaprvé je třeba být administrátor, což v případě útočníka znamená, že již plně ovládl počítač případně doménu. V takové situaci se již nebude zajímat o databázi hesel, neboť systém již natolik kompromitoval, že mu tato informace nepřinese žádná další privilegia v, pro něj otevřeném, systému. Jinými slovy v dnešní době je pro útočníka snazší nesnažit se prolomit autentizační protokoly ve své podstatě, ale pokusit se vstoupit do systému jiným způsobem, který autentizační mechanismus obejde. Pokud ovšem pro něj bude dostatečně zajímavé zcizit identitu někoho jiného a následně ji využít ve svůj prospěch, poté se cesta legitimního využití autentizačního protokolu stává relevantní a znalost jeho fungování tak sleduje zamýšlený cíl. V prvním případě se jedná o bezpečnostní problém informačních a operačních systémů všeobecně, neboť nalézt slabinu daných systémů při tak velkých objemech programového kódu, ze kterých se skládají je zcela jistě snazší, než tuto slabinu hledat v autentizačním protokolu, který je jen malou součástí celku a který je velmi dobře popsán, zdokumentován a „standardizován“. V případě druhém je cítit snaha tvůrců autentizačních protokolů aplikovat stále náročnější metody výpočtů, které sníží zranitelnost vůči útokům, proti kterým se daná opatření zavádějí.

8 Použitá literatura

[1] BAKER, K. UCLA Mathematics Department [online]. 1999 [cit. 2010-04-02]. Diffie-Hellman key exchange. Dostupné z WWW: <http://www.math.ucla.edu/~baker/40.1.99w/handouts/rev_DH/node1.html>.

[2] Berkeley Lab. Computer Protection Program [online]. 2004 [cit. 2010-02-25]. LAN Manager (LM) Hash Eradication. Dostupné z WWW: <http://www.lbl.gov/cyber/systems/lanman.html>.

[3] BITTO, O.. Bezpečná hesla?. *Connect!*. 2007, 2, s. 63.

[4] BITTO, O. Hesla, jejich solení a další autentizační koření. *Connect!*. 2007, 2, s. 61.

[5] BITTO, O.. Masarykova univerzita [online]. 2003 [cit. 2010-01-10]. HISTORIE KRYPTOLOGIE. Dostupné z WWW: <http://www.fi.muni.cz/usr/jkucera/pv109/2003/xbitto.htm>.

[6] CS Wikipedia [online]. 2009 [cit. 2010-01-14]. Hashovací funkce. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Hašovací_funkce>.

[7] E-government New Zealand [online]. 2004 [cit. 2010-01-10]. Best Practice Framework for Authentication. Dostupné z WWW: <http://www.e.govt.nz/services/authentication/library/docs/authentication-bpf/chapter5.html>.

[8] EGER, L.; BARTOŇKOVÁ, H. Centrum distančního vzdělávání [online]. 2004 [cit. 2010-04-05]. Příručka pro autory. Dostupné z WWW: <http://www.cdiv.upol.cz/www/autori_prirucka.htm>.

[9] EN Wikipedia [online]. 2010 [cit. 2010-01-18]. Message authentication code. Dostupné z WWW: <http://en.wikipedia.org/wiki/Message_authentication_code>.

[10] Federal Information Processing Standards Publication. FIPS PUBS [online]. 1993 [cit. 2010-04-10]. DATA ENCRYPTION STANDARD (DES). Dostupné z WWW: <http://www.itl.nist.gov/fipspubs/fip46-2.htm>.

[11] HÖLBL, M.; WELZER, T.; BRUMEN, B. Improvement of the Peyravian-Jeffrie´s user authentication protocol and password change protocol, Computer Communications 31 (2008) 1945-195.

[12] JELÍNEK, M. Autentizační tokeny v praxi. *IT SYSTEMS*. 2008, 10, s. 68.

[13] JÍCHA, R. *Interval.cz* [online]. 2005 [cit. 2010-02-10]. Salted hash - další krok ke zvýšení bezpečnosti. Dostupné z WWW: <http://interval.cz/clanky/salted-hash-dalsi-krok-ke-zvyseni-bezpecnosti/>.

[14] JOHANSSON, J., et al. Windows Server 2008 Security Resource Kit. Redmont, Washington : Microsoft Press, 2008. 484 s.

[15] JOHANSSON, J. TechNet Magazine [online]. 2006 [cit. 2010-03-15]. The Most Misunderstood Windows Security Setting of All Time. Dostupné z WWW: <http://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx>.

[16] LAMPORT, L. Research.microsoft.com [online]. 1981 [cit. 2010-02-20]. Password Authentication with Insecure Communication. Dostupné z WWW: <http://research.microsoft.com/en-us/um/people/lamport/pubs/password.pdf>.

[17] MALEČEK, L. Vysoká škola báňská [online]. 2008 [cit. 2010-03-08]. Šifrovací algoritmus DES. Dostupné z WWW: <http://homel.vsb.cz/~mor196/mal338.pdf>.

[18] Mendelova univerzita v Brně [online]. 2005 [cit. 2010-01-12]. Úvod do kryptologie. Dostupné z WWW: <http://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7021>.

[19] Mendelova univerzita v Brně [online]. 2005 [cit. 2010-01-23]. Digitální podpisy. Dostupné z WWW: <http://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7028>.

[20] Microsoft Corporation. MSDN [online]. 2010 [cit. 2010-02-21]. NTLM v1 Authentication. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/cc236699(PROT.10).aspx>.

[21] Microsoft Corporation. MSDN [online]. 2005 [cit. 2010-03-01]. Explained: Windows Authentication in ASP.NET 2.0. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa480475.aspx>.

[22] Microsoft Corporation. Technická podpora Microsoft [online]. 2005 [cit. 2010-03-02]. Povolení ověřování protokolem NTLM 2. Dostupné z WWW: <http://support.microsoft.com/kb/239869>.

[23] PŘIBYL, J. ČVUT Praha [online]. 2008 [cit. 2010-03-12]. Symetrické blokové šifry I. Dostupné z WWW: <http://www.comtel.cz/files/download.php?id=2570>.

[24] RSA Security. RSA Laboratories [online]. 2010 [cit. 2010-03-04]. What is Diffie-Hellman?. Dostupné z WWW: <http://www.rsa.com/rsalabs/node.asp?id=2248>.

[25] SILBERSCHATZ, A.; GALVIN, P.; GAGNE, G.. Operating System Concepts. USA : John Wiley & Sons. INC, 2005. 886 s. ISBN 0-471-69466-5.

[26] SkyNet, a.s. [online]. 2010 [cit. 2010-01-12]. Úvod do šifrování. Dostupné z WWW: <http://www.pgp.cz/index.php?l=cz&p=6&r=5>.

[27] STALLING, W. Operating Systems: Internals and Design Principles. 5th edition. [s.l.] : Prentice Hall, 2004. 832 s. ISBN 978-0131479548.

[28] ŠČUREK, J. Biometrické metody identifikace osob v bezpečnostní praxi [online]. 2008 [cit. 2010-02-11]. Dostupný z WWW: <http://www.fbi.vsb.cz/shared/uploadedfiles/fbi/biometricke_metody.pdf>.

[29] TANENBAUM, A. S. Modern Operating Systems. 3rd edition. [s.l.] : Prentice Hall, 2007. 1104 s. ISBN 978-0136006633.

[30] THE SEAN BLOG. *Layers (Defense in Depth Part 1)* [online]. 2007 [cit. 2009-12-11]. Dostupný z WWW: <http://blogs.technet.com/blogfiles/seanearp/WindowsLiveWriter/LayersDefenseinDepthPart1_B11E/CIA_triad.png>.

[31] TÖRNBLOM, H. Dominans [online]. 2007 [cit. 2010-03-21]. Skapa en LM Hash med Java. Dostupné z WWW: <http://www.bth.se/ht/bloxx.nsf/d6plinks/htom-7428z4>.

[32] WIDDENBECK S.; WATERS J.; BIRGET J.C.;BRODSKYI A.; MEMON N. ``PassPoints: Design and longitudinal evaluation of a graphical password system'', *International J. of Human-Computer Studies* (Special Issue on HCI Research in Privacy and Security), 63 (2005) 102-127.

[33] YANG, F.; WU, T.; HSU, M. Improvement of Hölbl et al. user authentication protocol and password change protocol. In KOTSIS, G., et al. Proceedings of the 11th International Conference on Information Integration and Web-based Application & Services (iiWAS2009). Malaysia : ACM, 2009. s. 756. ISBN 978-1-60558-660-1.

9 Přílohy

9.1 Diffie – Hellman protokol pro výměnu klíčů

Diffie-Hellman protokol [24][1] byl vyvinut pány Whitfieldem Diffie a Martinem Hellmanem v roce 1976 a publikován v díle „New Directions in Cryptography“. Protokol umožňuje dvěma uživatelům vyměnit si tajný klíč přes nezabezpečené médium. Systém, na kterém je tato výměna postavena, lze charakterizovat takto:

Protokol obsahuje dva systémové parametry p a q. Oba dva jsou veřejné a mohou tak být použity účastníky výměny. Parametr p je prvočíslo (velké) a parametr g (nazývaný „generátor“) je celé číslo, které je menší než p a zároveň splňuje tu vlastnost, že pro každé číslo n z intervalu $< 1 ; p\text{-}1 >$ existuje takový exponent k pro g takový, že:

$$n = g^k mod\ p$$

Předpokládejme, že Alice a Bob chtějí sdílet tajný klíč a přenést jej za pomoci Diffie-Hellmanova protokolu. Jejich postup bude následující:

1. Alice si zvolí náhodné tajné číslo X_A a Bob si zvolí náhodné tajné číslo X_B. Oba parametry jsou celá čísla.
2. Následně oba vypočítají své veřejné hodnoty za použití parametrů p a q. Alice použije vztah $Y_A = g^{X_A} mod\ p$ a Bob vztah $Y_B = g^{X_B} mod\ p$.
3. Následně si vymění svá veřejná čísla Y_A a Y_B.
4. Pro získání tajného klíče k Alice vypočítá vztah $g^{X_A X_B} = (Y_B)^{X_A}\ mod\ p$ a Bob vypočítá $g^{X_B X_A} = (Y_A)^{X_B}\ mod\ p$.
5. Jelikož $g^{X_A X_B} = g^{X_B X_A} = k$, Alice i Bob sdílejí stejný klíč.

Příklad:

- Zvolíme $p = 11$, $g = 2$, $X_A = 9$, $X_B = 4$.
- Tedy $Y_A = 2^9 \pmod{11}$, tedy $Y_A = 6$.
- Obdobně $Y_B = 2^4 \pmod{11}$, tedy $Y_B = 5$.
- Čísla Y_A i Y_B zveřejníme.

Tajný klíč k vypočítá Alice dle vztahu $k = g^{X_A X_B} = (Y_B)^{X_A}\ mod\ p$, tedy $k = 5^9 \pmod{11} = 9$.

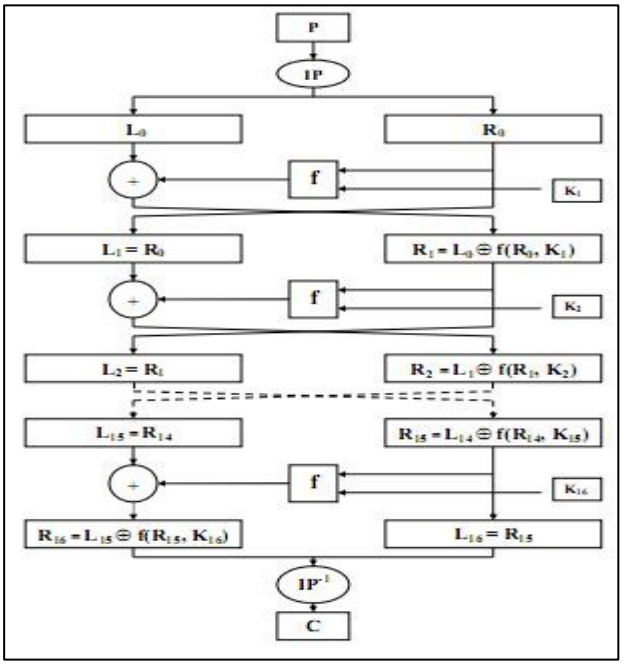
Tajný klíč k vypočítá Bob dle vztahu $k = g^{X_B X_A} = (Y_A)^{X_B}\ mod\ p$, tedy $k = 6^4 \pmod{11} = 9$.

Z výše uvedeného popisu je vidět nevýhoda tohoto protokolu. Je zranitelný vůči útoku man-in-the-middle. V tomto případě narušitel Tom zachytí veřejnou hodnotu Alice Y_A a zašle Bobovi svou veřejnou hodnotu. Když Bob vyšle svou veřejnou hodnotu Y_B, Tom tuto zachytí a nahradí ji svou, kterou zašle Alici. Tom a Alice se tedy dohodnou na sdíleném tajném klíči, zatímco Tom a Bob se dohodnou na jiném tajném klíči. Na základě tohoto bude Tom schopen jednoduše dešifrovat zprávu zaslanou Alicí případně Bobem, přečte si ji a zmodifikuje ještě předtím, než ji opět zašifruje příslušným klíčem a zašle ji protistraně [24].

Zranitelnost tohoto řešení tak spočívá v tom, že v rámci Diffie – Hellmanova protokolu se mezi sebou neautentizují komunikující strany. Možným řešením je zahrnout použití digitálních podpisů, případně jiné varianty tohoto protokolu.

9.2 Data Encryption Standard - DES

Algoritmus DES je symetrická bloková šifra [17][23][10]. Z této definice je patrné, že pro šifrování a dešifrování využívá stejný klíč (symetrický) a že zprávu šifruje po určitých blocích (konkrétně 64 bitů). Ke své činnosti využívá tzv. Fistelovu síť, která určuje, jak se provádí jeden krok šifrování a určuje i počet opakování tohoto kroku. V případě šifry DES se jedná o 16 opakování. Délka klíče u šifry DES činí 56 bitů[13].



Obrázek 23 - Schéma šifrovacího algoritmu DES [23]

[13] Někdy se uvádí 64 bitů, kdy se za pomoci kompresní permutace vybere 56 bitů a zbylé bity slouží jako paritní (bezpečnostní) a při vlastním šifrování jsou ignorovány.

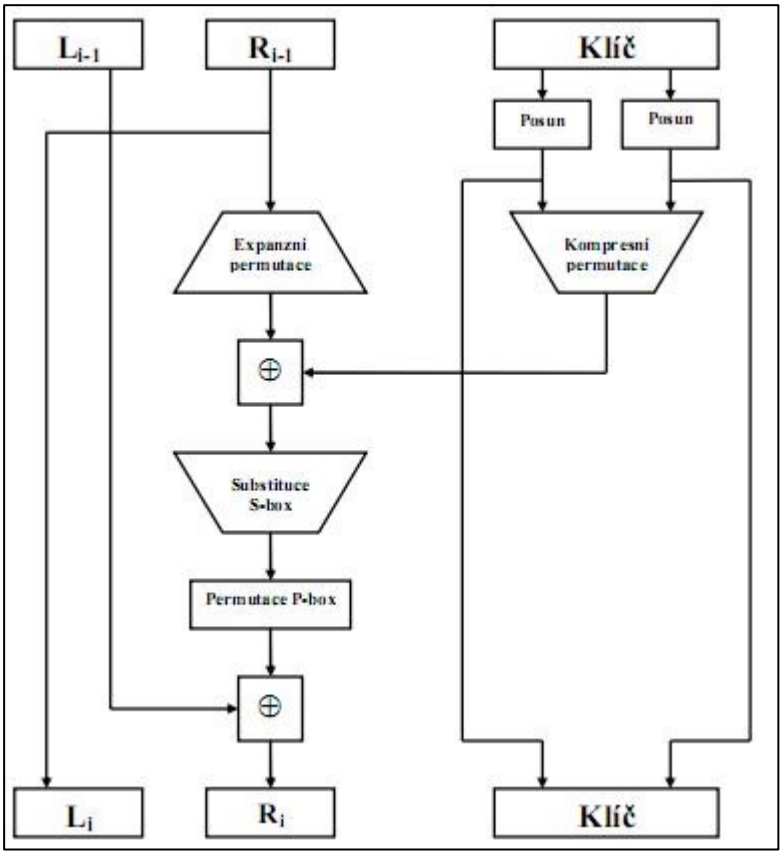
Na Obrázku 23 je uvedeno základní schéma šifry DES. Prvním krokem šifrování je úvodní permutace bloku zprávy fixní permutací uvedenou na Obrázku 24 a to tak, že vstupní osmibitová hodnota slouží jako index do této tabulky (4 bity jsou indexem řádku a 4 bity jsou indexem sloupce.) a na výstup se dává hodnota nalezena v této tabulce v průsečíku daného řádku a sloupce.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Obrázek 24 - Úvodní permutace DES [23]

Druhý krok šifrování spočívá v aplikaci 16 opakování kroku, který pro svou činnost využívá klíč dlouhý 48 bitů, který se generuje pro každé opakování jiný z úvodního 56 bitového klíče. Posledním krokem je aplikace inverzní permutace, po níž dostáváme zašifrovaný blok zprávy.

Konkrétní podoba jednoho kroku DES vypadá následovně:



Obrázek 25 – Jeden krok šifry DES [23]

Z obrázku je patrno, že blok zprávy je rozdělena na pravou a levou polovinu. 32 bitů pravé poloviny zkopírujeme do dalšího opakování jako novou levou polovinu. Dále pravou polovinu za pomoci expanzní permutace rozšíříme na 48 bitů. Expanzní permutace je obdobná úvodní permutaci, využívá indexy řádků a sloupců, přičemž rozšíření je realizováno zopakováním některých hodnot.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Obrázek 26 - Expanzní permutace [23]

Dalším krokem je spojení výsledku expanzní permutace s klíčem daného opakování (48 bitů dlouhý) pomocí funkce XOR. Jejím výsledkem je 48bitová hodnota. Ta vstupuje do celkem osmi S - boxů, jejichž výstupem je 32bitová hodnota. S-box provádí substituci, čili zamění bity, které jsou na vstupu tak, že na výstup dá jinou kombinaci bitů, která nemusí být co do počtu nutně shodná se vstupní kombinací. Konkrétně tedy je do S-boxu přivedeno 6 bitů, přičemž výstupem jsou pouze 4 bity. Vstup je opět indexem tabulky a to tak, že první a poslední bit je index řádku a prostřední 4 bity jsou indexem sloupce. Tabulky všech osmi S-boxů jsou pevně dány.

Následujícím krokem je permutace výsledku za pomoci P-boxu, který má podobnou funkci jako S-box, tedy že zaměňuje bity, ovšem na rozdíl od S-boxů má vstup i výstup stejnou délku (P-box tedy pouze mění pořadí bitů).

Posledním krokem v daném opakování je spojení původní levé a upravené pravé strany pomocí funkce XOR, čímž je připravena nová pravá strana do dalšího opakování Fistelovi sítě.

Co se týče generování klíčů pro jednotlivá opakování schematicky naznačenému na Obrázku 25. Klíč je nejprve rozdělen na dvě poloviny o délce 28 bitů. Každou polovinu následně posuneme bitově doleva. Pro opakování č. 1, 2, 9 a 16 se posouvá o 1 bit, pro ostatní o 2 bity (posunem zároveň dostaneme i nové poloviny pro následující opakování). Pomocí kompresní permutace zkrátíme klíč z 56 bitů na 48 bitů za využití těchto hodnot:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Obrázek 27 - Permutace pro generování klíčů [23]

Dešifrování zprávy se děje pouhým otočením pořadí kroků při šifrování. Tedy začíná se od posledního opakování č. 16 a bitové posuny s klíči se dějí doprava. Výhodou tohoto řešení je možnost využití stejného hardware pro obě operace.