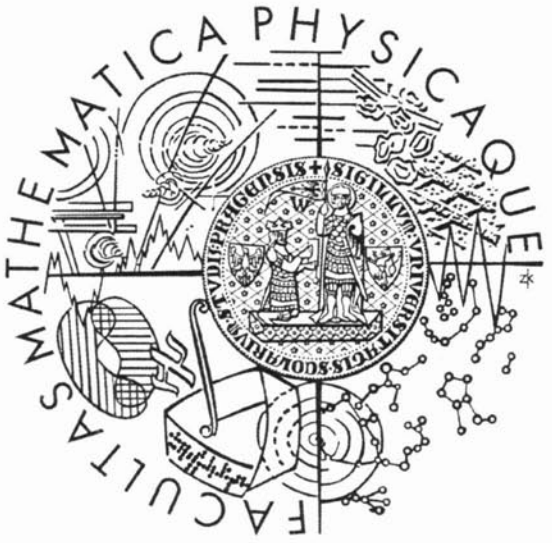
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Miroslava Prater

Protokoly pro dohodu na klíči

Katedra algebry

Vedoucí bakalářské práce: Doc. RNDr. Jiří Tůma, DrSc., Katedra algebry
Studijní program: Matematika
Studijní obor: Matematické metody informační bezpečnosti

2007

Název práce: Protokoly pro dohodu na klíči
Autor: Miroslava Prater
Katedra (ústav): Katedra algebry
Vedoucí bakalářské práce: Doc. RNDr. Jiří Tůma, DrSc.
e-mail vedoucího: tuma@karlin.mff.cuni.cz

Abstrakt: V práci předkládám stručný přehled typů protokolů pro vytvoření klíče, které jsou nezbytné pro bezpečnost elektronické komunikace. V první kapitole uvádím základní pojmy. V další kapitole je popsán příklad protokolu pro dohodu na klíči, který využívá symetrické šifrování; následující kapitola pojednává o protokolech pro transport klíče, které jsou taktéž založeny na symetrickém šifování, dále pojednává o obnově klíče a transportu klíče bez předem sdíleného klíče a nakonec o protokolech založených na využití důvěryhodného serveru. Ve čtvrté kapitole jsou popsány dva protokoly pro transport klíče založené na šifrování s veřejným klíčem a jeden hybridní protokol. V poslední kapitole se věnuji protokolům založeným na Diffie-Hellmanově dohodě na klíči a jejich variantám s implicitně certifikovanými a samocertifikovanými klíči.

Klíčová slova: kryptografické protokoly, vytvoření klíče, dohoda na klíči, transport klíče, autentizace.

Title: Key establishment protocols
Author: Miroslava Prater
Department: Department of Algebra
Supervisor: Doc. RNDr. Jiří Tůma, DrSc.
Supervisor's e-mail address: tuma@karlin.mff.cuni.cz

Abstract: In this work, I will present a concise overview of different types of key establishment protocols essential for secure electronic communication. I start with an introduction to the topic and an outline of the basic terms and definitions for terminology. We then go on to look at an example key agreement protocol which uses symmetric encryption. The following chapter focuses on key transport protocols which use symmetric encryption, key update and key transport without a priori shared key, and server-based protocols. In the fourth chapter, we will move on to examine two key transport protocols using public-key encryption and one hibrid protocol. The last chapter is devoted to Diffie-Hellman key agreement and its related protocols and their variants with implicitly certified and self-certified keys.

Keywords: cryptographic protocols, key establishment, key agreement, key transport, authentication.

Na tomto místě bych chtěla poděkovat panu Doc. Jiřímu Tůmovi za hodnotné rady a svému manželovi za podporu při tvorbě této práce.

Prohlašuji, že jsem svou bakalářskou práci napsala samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 3. srpna 2007 Miroslava Prater

Obsah

Úvod

Na úvod terminologická poznámka. Protokoly, kterými se v této práci zabývám, jsou v anglické terminologii označovány jako *key establishment protocols*. Tyto protokoly se dělí do dvou velkých skupin s anglickými názvy *key agreement protocols* a *key transport protocols*. V dalším textu tyto pojmy překládám do češtiny jako „protokoly pro vytvoření klíče" (*key establishment protocols*), „protokoly pro dohodu na klíči" (*key agreement protocols*) a „protokoly pro transport klíče" (*key transport protocols*). V dalším textu má tedy výraz „dohoda na klíči" užší význam, než jaký by odpovídal zadání práce. S vytvořením klíče je úzce spjata autentizace, takže někdy se tato skupina kryptografických protokolů v anglické literatuře označuje jako *protocols for authentication and key establishment*, což by v terminologii odpovídající této práci bylo označováno jako „protokoly pro autentizaci a vytvoření klíče". Podrobněji se terminologii a definicím základních pojmů věnuje první kapitola.

Protokoly pro vytvoření klíče, použité v této práci, jsou rozděleny do čtyř kapitol podle toho, zda se jedná o protokoly pro transport klíče nebo protokoly pro dohodu na klíči, a podle toho, zda využívají symetrické šifrování či šifrování s veřejným klíčem. Jednotlivé protokoly či skupiny protokolů založených na stejném principu jsou pak v každé kapitole odděleny do zvláštních sekcí a podsekcí. U některých protokolů jsou také uváděny slabiny, případně konkrétní útok a opatření, která mají zajistit jak takovému útoku předejít.

Otázka bezpečnosti je v této práci zmíněna jen okrajově; jde spíše o rámcový přehled metod, které jsou použity v daných protokolech a na nich také popsány. Rozsah je víceméně stejný jako v knize [1], ze které jsem protokoly převzala. Útoky, které jsou zmíněny u některých protokolů, jsou převzaty z knihy [3].

Kapitola 1

Základní klasifikace a důležité pojmy

1.1 Šifry, schémata, funkce

Definice v tomto oddíle jsou převzaty z knih [1] a [2].

Symetrická šifra (*Symmetric encryption*) Při použití symetrické šifry v komunikaci si musí nejprve komunikující strany předem vyměnit tajný šifrovací klíč, který budou sdílet. Strana A pak zašifruje posílanou zprávu tímto tajným klíčem a strana B použije týž klíč při aplikování dešifrovacího algoritmu na obdrženou zprávu.

Symetrická šifra sestává ze tří množin - množiny klíčů $\mathcal{K}$, množiny zpráv $\mathcal{M}$ a množiny šifrových textů $\mathcal{C}$, spolu se dvěma algoritmy:

1. Šifrovací algoritmus, který vezme zprávu $m \in \mathcal{M}$ a klíč pro zašifrování $K \in \mathcal{K}$ a jeho výstupem je prvek $c \in \mathcal{C}$, který je definován jako $c = E_K(m)$. Výstup algoritmu je pro dané K a m jednoznačně určený - výsledkem nemohou být ruzná c pro stejné m při pevně zvoleném klíči K.

2. Dešifrovací algoritmus, jehož vstupem je $c \in \mathcal{C}$ a klíč pro dešifrování $K \in \mathcal{K}$ a výstupem je prvek $m \in \mathcal{M}$ takový, že $m = D_K(c)$. Přitom platí, že $D_K(E_K(m)) = m$.

V kryptografických protokolech se využívá bloková šifra (tj. šifruje se a dešifruje po blocích pevné délky obvykle 8 B), protože zprávy protokolu jsou krátké. V dnešní době je doporučeno používat symetrický šifrovací algoritmus AES (*Advanced Encryption Standard*) s délkou klíče 128, 192 nebo 256 bitů.

Šifra s veřejným klíčem (*Public-key encryption*) Narozdíl od předešlé, šifra s veřejným klíčem používá dvojici šifrovacích klíčů - veřejný a soukromý. Přitom veřejný klíč je použit pro zašifrování a soukromý pak pro dešifrování.

Nejznámějším algoritmem pro šifru s veřejným klíčem je RSA, doporučená délka klíče je alespoň 1024 bitů, i když často se používají i delší (2048 či 4096) podle předpokládaných možností potencionálního útočníka. Za zmínku stojí také algoritmus ECC (*Eliptic Curve Cryptography*); pro srovnání - 1024 bitů dlouhý RSA klíč odpovídá co do bezpečnosti 160 bitů dlouhému klíči ECC, výpočetní náročnost je srovnatelná (podle [2]).

Šifrovací algoritmy použité v protokolech v této práci jsou zejména RSA, ElGamal a Rabinova šifra s veřejným klíčem, které jsou stručně popsány v následujících třech pododdílech a jsou převzaty z knihy [1].

1.1.1 RSA

Generování veřejného klíče a jemu odpovídajícího soukromého klíče pro RSA šifru s veřejným klíčem provádí entita A následujícím způsobem:

1. Vygeneruje dvě náhodná velká, navzájem různá prvočísla p, q, řádově stejně velká.
2. Spočítá $n = pq$ a $\phi = (p-1)(q-1)$.
3. Vybere náhodné číslo e, $1 < e < \phi$, takové, že $GCD(e, \phi) = 1$ (GCD značí největší společný dělitel).
4. Použije rozšířený Eukleidův algoritmus (EEA) pro výpočet jednoznačně určeného čísla d, $1 < d < \phi$, takového, že $ed \equiv 1 \pmod{\phi}$.
5. Veřejný klíč entity A je (n, e), její soukromý klíč je d.

Číslo e se nazývá šifrovací exponent, d je dešifrovací exponent. Číslo n se nazývá modulus.

Šifrovací algoritmus:

I. Při šifrování B postupuje takto:

1. Obdrží (n, e) - veřejný klíč strany A.
2. Převede zprávu na číslo m z intervalu $[0, n-1]$.
3. Spočítá $c = m^e \bmod n$.
4. Pošle šifrový text c entitě A.

II. Při dešifrování postupuje A takto:

Použije soukromý klíč d k získání zprávy m tak, že spočítá $m = c^d \bmod n$.

Důkaz, že dešifrování funguje, následuje.
Z $ed \equiv 1 \bmod \phi$ vyplývá existence takového k, že $ed = 1 + k\phi$.

1. Pokud $GCD(m, p) = 1$, pak podle malé Fermatovy věty platí $m^{p-1} \equiv 1 \pmod p$. Umocněním obou stran kongruence exponentem $k(q-1)$ a následným vynásobením m dostáváme kongruenci $m^{1+k(p-1)(q-1)} \equiv m \pmod p$, čili $m^{1+k\phi} \equiv m \bmod p$.

2. Pokud $GCD(m, p) = p$, pak kongruence $m^{1+k\phi} \equiv m \bmod p$ opět platí, neboť obě strany jsou kongruentní s 0 mod p.

Tedy v obou případech platí $m^{ed} \equiv m \bmod p$.

Z obdobných důvodů platí též kongruence $m^{ed} \equiv m \bmod q$.

Jelikož jsou p, q navzájem různá prvočísla, vyplývá z posledních dvou kongruencí pomocí Čínské věty o zbytcích $m^{ed} \equiv m \bmod n$. A tedy platí $c^d \equiv (m^e)^d \equiv m \bmod n$.

Problém získání otevřeného textu m z šifrového textu c při známých hodnotách (n, e) se nazývá RSA problém (RSAP). Dosud není znám žádný efektivní algoritmus pro tento problém.

Prvočísla, ze kterých se skládá modulus, by měla být stejné bitové délky, takže například pro 1024-bitový modulus by měla být dlouhá 512 bitů.

1.1.2 ElGamal

Při generování veřejného a jemu odpovídajícího soukromého klíče postupuje každá entita A takto:

1. Vygeneruje náhodné velké prvočíslo p a generátor α multiplikativní grupy $\mathbb{Z}_p^*$.
2. Vybere náhodné číslo a, $1 \leq a \leq p-2$, a spočítá $\alpha^a \bmod p$.
3. Veřejný klíč entity A je (p, α, α^a).

Šifrovací algoritmus:

I. Při šifrování postupuje B následujícím způsobem:

1. Obdrží (p, α, α^a) - veřejný klíč strany A.

2. Převede zprávu na číslo m z intervalu $[0, p-1]$.

3. Vybere náhodné číslo k, $1 \leq k \leq p-2$.

4. Spočítá $\gamma = \alpha^k \bmod p$ a $\delta = m \cdot (\alpha^a)^k \bmod p$.

5. Pošle šifrový text $c = (\gamma, \delta)$ entitě A.

II. Při dešifrování c postupuje A takto:

1. Použije soukromý klíč a k výpočtu $\gamma^{p-1-a} (= \gamma^{-a} = \alpha^{-ak}) \bmod p$.

2. Získá zprávu m tak, že spočítá $(\gamma^{-a}) \cdot \delta \bmod p$.

Důkaz, že dešifrování funguje, je následující: $\gamma^{-a} \cdot \delta \equiv \alpha^{-ak} \cdot m\alpha^{ak} \equiv m \pmod{p}$.

1.1.3 Rabinova šifra s veřejným klíčem

Při generování veřejného a jemu odpovídajícího soukromého klíče postupuje každá entita A takto:

1. Vygeneruje dvě náhodná velká, navzájem různá prvočísla p, q, řádově stejně velká.

2. Spočítá $n = pq$.

3. Veřejný klíč entity A je n, její soukromý klíč je (p, q).

Šifrovací algoritmus:

I. Při šifrování zprávy m postupuje entita A následovně:

1. Obdrží n - veřejný klíč entity A.

2. Převede zprávu na číslo m z intervalu $[0, n-1]$.

3. Spočítá $c = m^2 \bmod n$.

4. Pošle šifrový text c entitě A.

II. Při dešifrování c postupuje A takto:

1. Nalezne druhé odmocniny m_1, m_2, m_3 a m_4 z c mod n.

2. Poslaná zpráva je jedna z těchto čtyř odmocnin a A se nějakým způsobem rozhodne pro jednu z nich.

1.2 Protokol a s ním související pojmy

Definice v této sekci jsou převzaty z knihy [1].

Protokol je algoritmus, kterého se účastní více stran, definovaný posloupností kroků přesně specifikujících akci požadovanou dvěma či více stranami za účelem docílit určeného úkolu.

Vytvoření klíče (*key establishment*) je proces nebo protokol, jehož pomocí se sdílené tajemství stává dostupným dvěma či více stranám pro pozdější použití.

Proces vytvoření klíče se může mimo jiné rozdělit na transport klíče a dohodu na klíči.

Protokol pro transport klíče (*key transport protocol*) je technika vytvoření klíče, kde jedna strana vytvoří nebo jinak získá tajnou hodnotu a bezpečně ji předá druhé či dalším stranám.

Protokol pro dohodu na klíči (*key agreement protocol*) je technika vytvoření klíče, ve které je sdílené tajemství odvozeno dvěma či více stranami jako funkce informace, kterou přispěla každá ze stran, v ideálním případě tak, že žádná ze stran nemůže předem určit výslednou hodnotu klíče.

Následující rozdělení protokolů pro vytvoření klíče je podle počátečního nastavení - výsledný klíč může být při každém běhu protokolu stejný nebo je pokaždé vypočítán jiným způsobem, a je tedy pokaždé jiný.

Protokol pro predistribuci klíče (*key pre-distribution protocol*) je protokol pro vytvoření klíče, jehož pomocí jsou výsledné vytvořené klíče zcela určené *a priori* počátečním klíčovým materiálem narozdíl od **protokolu pro dynamické vytvoření klíče** (*dynamic key establishment protocol*), kde vytvořené klíče závisí na pozdějších výpočtech. Dynamické vytvoření klíče se někdy označuje jako vytvoření klíče pro danou relaci (*session key establishment*).

Dynamické vytvoření klíče je technika, kterou využívají jak protokoly pro transport klíče, tak protokoly pro dohodu na klíči; predistribuce klíče je využita pouze v protokolech pro dohodu na klíči.

Mnoho protokolů pro vytvoření klíče vyžaduje centralizovanou či důvěryhodnou stranu kvůli počátečnímu nastavení systému a/nebo on-line akcím. Jsou to zejména třetí důvěryhodná strana, důvěryhodný server, ověřovací server, centrum pro distribuci klíče (KDC), centrum pro přenos klíče (KTC) a certifikační

...FYZ. FAKULTY
...oddělení
...ská 83
...5 Praha 3

autorita (CA), jejichž rolemi a funkcí se nyní nebudeme podrobně zabývat.

1.3 Kryptografické vlastnosti

Kryptografické algoritmy mají zajistit, aby měl protokol některé kryptografické vlastnosti z následujícího stručného přehledu. Údaje v tomto oddíle jsou převzaty z knihy [3].

Důvěrnost, utajení (*Confidentiality*) Kryptografické klíče a jiná data jsou dostupná pouze oprávněným stranám.

Autentizace entity, identifikace (*Entity authentication, identification*) Potvrzení totožnosti (identity) entity (např. osoby, terminálu, kreditní karty, atd.); proces, při kterém jedna strana prokazuje svou totožnost straně druhé a zároveň potvrzuje bezprostředně předchozí aktivitu či účast na komunikaci.

Autentizace dat (*Data origin authentication*) zaručuje jedné straně původ dat poslaných druhou stranou (tedy že data nebyla podstrčena nějakou střetí stranou) a **integritu dat** (*data integrity*) (tedy že data nebyla změněna neoprávněnou osobou).

Nepopiratelnost (*Non-repudiation*) Komunikující strany nemohou popřít odeslání či podepsání dat, která odeslaly či podepsaly.

Potvrzení přijetí klíče (*Key confirmation*) Jedna ze stran je ujištěna, že druhá (i neidentifikovaná) strana již obdržela daný tajný klíč.

Autentizace klíče (*Key authentication*) Jedna ze stran je ujištěna, že žádná jiná strana kromě výslovně určené druhé strany (resp. dalších) nemůže získat přístup k danému tajnému klíči.

Jelikož autentizace klíče nezávisí na skutečnosti, zda onen klíč opravdu druhá strana má, označuje se autentizace klíče přesněji jako **implicitní** autentizace klíče (*implicit key authentication*). Pokud je zaručena jak autentizace klíče, tak potvrzení přijetí klíče, pak je dosaženo **explicitní** autentizace klíče (*explicit key authentication*).

1.4 Klasifikace útoků

Útoky na kryptografické protokoly se dají široce rozdělit na **pasivní útok** (*passive attack*) - takový, kde útočník jednoduše nahrává data a poté je analyzuje a nijak

nenarušuje běh protokolu - a na **aktivní útok** (*active attack*) - takový, kde útočník modifikuje posílané zprávy či vkládá zprávy do komunikace. Podkladem pro tuto sekci byla kniha [3].

Všechny následující typy útoků kromě prvního vyžadují, aby byl útočník aktivní.

Odposlech (*Eavesdropping*) Útočník se zmocní informace poslané v protokolu.

Tento typ útoku bývá samozřejmou součástí většiny útoků na protokoly, proto by měl být na paměti při budování či používání každého protokolu.

Modifikace (*Modification*) Útočník změní informaci poslanou v protokolu.

Přehrání (*Replay*) Útočník zaznamená, nahraje informaci zachycenou v průběhu protokolu a poté ji odešle tomu samému či jinému účastníkovi v pozdější fázi běhu protokolu (pokud je to možné).

V dalších kapitolách budu používat termín *replay attack* nebo jen *replay*, neboť český ekvivalent není vžitý.

Předstih, předběhnutí (*Preplay*) Útočník se zapojí do běhu protokolu dříve než oprávněný účastník.

Odraz (*Reflection*) Útočník pošle zprávu protokolu zpátky tomu, kdo ji odeslal.

Zamítnutí služby (*Denial of service*) Útočník zabrání oprávněným účastníkům dokončit protokol.

Přepisovací útok (*Typing attack*) Útočník nahradí pole určitého typu v zašifrované zprávě protokolu polem jiného typu.

Tento druh útoku je popsán u Otwayova-Reesova protokolu (viz 3.6).

Kryptoanalýza (*Cryptanalysis*) Útočník získá nějakou užitečnou informaci z průběhu protokolu, která mu pomůže v kryptoanalýze.

Manipulace s certifikáty (*Certificate manipulation*) Útočník vybere nebo změní některé údaje z certifikátu, aby mohl zaútočit na jeden či více běhů protokolu.

Vzájemné působení mezi protokoly (*Protocol interaction*) Útočník zvolí nový protokol, který souvisí s nějakým známým protokolem.

1.5 Aktuálnost

Tato sekce se zabývá problémem aktuálnosti (*freshness*) dat v protokolech a způsoby, jak jí docílit. Opět jsem čerpala z knihy [3].

Časové razítko (*Timestamp*) Odesílatel zprávy přidá nynější čas ke zprávě v moment, kdy ji odesílá. Tento čas je pak zkontrolován příjemcem při přijmutí zprávy srovnáním s místním časem. Pokud je obdržené časové razítko přijato v akceptovatelném časovém rámci, pak je zpráva považována za čerstvou. Potíž vužívání časového razítka je v tom, že jsou vyžadovány synchronizované hodiny u všech účastníků protokolu a musí být spolehlivě udržovány.

Nonce Tento termín nemá v češtině ustálený ekvivalent, proto jej budu užívat vanglickém znění. Znamená něco jako „číslo použité pouze jednou" (*number used once*). Příjemce zprávy vygeneruje náhodné číslo (*nonce*) a pošle jej odesílateli. Toto *nonce* je pak navráceno se zprávou po vykonání nějakého kryptografického algoritmu. Příjemce zkontroluje *nonce* v odpovědi a usoudí, že zpráva je čerstvá, protože nemohla být vytvořena předtím, než bylo *nonce* vygenerováno. Důležitá je kvalita *nonce*, neboť pokud je *nonce* předvídatelné, pak může být platná odpověď obdržena v předstihu a odeslána později (*replay attack*). Za dostatečně dlouhé *nonce* se považuje číslo delší než 16 B (podle [2]). Někdy se *nonce* vytváří tak, že se složí zedvou částí - náhodného čísla a data a času.

Čítač (*Counter*) Odesílatel a příjemce udržují synchronizovaný čítač, jehož hodnota je vždy odeslána spolu se zprávou a následně zvýšena. Nevýhodou čítače je, že daná informace musí být udržována s každým potencionálním partnerem.

1.6 Digitální podpis

Údaje v této sekci a jejích podsekcích jsou převzaty z knihy [1].

Digitální podpis (*Digital signature*) Tento mechanismus zajišťuje jedné straně nepopiratelnost dat. Skládá se ze dvou částí - **podepsání** (*signing*) a **ověření** (*verification*).

Podpisový algoritmus sestává z množiny zpráv $\mathcal{M}$ a z množiny podpisů $\mathcal{S}$ (podpis je zpravidla binární řetězec pevně stanovené délky).

Dále zahrnuje podpisovou transformaci nebo funkci (*signing transformation*) S_A pro entitu A, která zobrazuje $\mathcal{M}$ do $\mathcal{S}$ a je držena entitou A v tajnosti a používána k podepsání zpráv z $\mathcal{M}$. Při procesu podepsání zpávy $m \in \mathcal{M}$ postupuje A takto:

1. Spočítá $s = S_A(m)$, s je podpis zprávy m.
2. Odešle dvojici (m, s).

V_A je ověřovací funkce (*verification transformation*) z $\mathcal{M} \times \mathcal{S}$ (množiny dvojic (m, s)) do množiny $\{true, false\}$, je veřejná a užívají ji ostatní entity pro ověření podpisů entity A. Proces ověření, zda podpis s zprávy m vytvořila entita A, je následující:

1. Ověřovatel (*verifier*) si opatří ověřovací funkci V_A entity A a spočítá $u = V_A(m, s)$.
2. Pokud $u = true$, přijme podpis jako podpis entity A. Pokud $u = false$, zamítne jej.

Existuje řada podpisových a ověřovacích algoritmů, které jsou veřejné, ale jsou charakterizovány klíčem. Takže podpisový algoritmus S_A entity A je určen tajným klíčem k_A, ověřovací algoritmus V_A je určen veřejným klíčem l_A.

S_A a V_A musí splňovat tyto požadavky: s je platný podpis zprávy m entity A, právě když $V_A(m, s) = true$, a pro jiné entity je výpočetně velmi těžké pro nějaké $m \in \mathcal{M}$ nalézt takové $s \in \mathcal{S}$, že $V_A(m, s) = true$.

Digitální podpis má dvě základní třídy. Jedna zahrnuje **digitální podpisy s obnovou zprávy** (*digital signatures with message recovery*), které používají tzv. redundanční funkci (*redundancy function*) (viz následující podsekce). Druhá třída zahrnuje **digitální podpisy s dodatkem** (*digital signatures with appendix*), které používají hašovací funkci.

Jako příklad digitálního podpisu s obnovou zprávy zde uvedu RSA podpis a jako příklad podpisu s dodatkem popíši ElGamalův podpis.

1.6.1 RSA podpis

Tento podpisový algoritmus umožňuje zpětné získání původní zprávy z podpisu - obnovu zprávy (*message recovery*), tzn. není potřeba znalosti původní zprávy při ověřování.

Použití redundanční funkce R u podpisu s obnovou zprávy má zabránit útočníkovi, aby mohl generovat platné podpisy i přes neznalost tajného klíče k pro podpisovou transformaci $S_{A,k}$.

Množina $\mathcal{M}_S$ je množina elementů, na které je aplikována podpisová funkce. Funkce R je bijekce z $\mathcal{M}$ do $\mathcal{M}_S$, podpisová transformace $S_{A,k}$ je bijekce z $\mathcal{M}_S$ do $\mathcal{S}$. Dále označíme $\mathcal{M}_R = Im(R)$. Množina $\mathcal{R}$ je indexující množina pro podpisové funkce. Musí platit $\mathcal{M}_R \subseteq \mathcal{M}_S$ z důvodu, který je zřejmý z následujícího příkladu volby redundanční funkce.

Nechť $\mathcal{M} = \{m : m \in \{0,1\}^n\}$ a $\mathcal{M}_\mathcal{S} = \{t : t \in \{0,1\}^{2n}\}$. Definujeme $R : \mathcal{M} \to \mathcal{M}_\mathcal{S}$ předpisem $R(m) = m||m$, kde $||$ je konkatenace. Takže $\mathcal{M}_R = \{m||m : m \in \mathcal{M}\} \subseteq \mathcal{M}_\mathcal{S}$. Pro velké hodnoty n je zlomek $|\mathcal{M}_R|/|\mathcal{M}_\mathcal{S}| = (1/2)^n$ zanedbatelně malý. Takže náhodná volba podpisu $s^* \in \mathcal{S}$ zaručí pouze zanedbatelně malou pravděpodobnost, že $V_A(s^*) \in \mathcal{M}_R$.

Nyní přejdeme k samotnému podpisu. Nejprve je nutné, aby entita A vygenerovala své RSA klíče - veřejný klíč a jemu odpovídající soukromý klíč. Postup je stejný jako u šifry RSA, veřejný klíč entity A je (n, e), její soukromý klíč je d.

Prostor zpráv $\mathcal{M}$ a prostor podpisů $\mathcal{S}$ jsou oba $\mathbb{Z}_n = \{0, 1, 2, ..., n-1\}$. Nyní může proběhnout algoritmus pro podepsání a ověření:
1. Entita A podepíše zprávu $m \in \mathcal{M}$ takto:
   (a) Spočítá $\widetilde{m} = R(m)$ – číslo z intervalu $[0, n-1]$, R je redundanční funkce.
   (b) Spočítá $s = \widetilde{m}^d \bmod n$.
   (c) s je podpis zprávy m strany A.
2. Nějaká jiná entita B ověří tento podpis a zároveň z podpisu získá zprávu m:
   (a) Obdrží veřejný klíč (n, e) entity A.
   (b) Spočítá $\widetilde{m} = s^e \bmod n$.
   (c) Ověří, zda $\widetilde{m} \in \mathcal{M}_R$, kde $\mathcal{M}_R = Im(R)$; pokud ne, podpis zamítne.
   (d) Získá zprávu $m = R^{-1}(\widetilde{m})$, kde R^{-1} je funkce inverzní k R.

Důkaz, že ověření podpisu opravdu funguje, je následující:

Je-li s podpis zprávy m, pak $s \equiv \widetilde{m}^d \bmod n$, kde $\widetilde{m} = R(m)$. Z $ed \equiv 1 \bmod \phi$ vyplývá $s^e \equiv \widetilde{m}^{ed} \equiv \widetilde{m} \bmod n$. Nakonec, $R^{-1}(\widetilde{m}) = R^{-1}(R(m)) = m$.

1.6.2 ElGamalův podpis

Toto podpisové schéma generuje digitální podpisy s dodatkem ze zpráv - binárních řetězců libovolné délky. Podpisová schémata s dodatkem jsou taková, která vyžadují, aby byla zpráva vstupem ověřovacího algoritmu, a vyžadují použití hašovací funkce $h : \{0,1\}^* \to \mathbb{Z}_p$, kde p je velké prvočíslo.

Nejprve musí entita A vytvořit veřejný a k němu příslušný soukromý klíč. Při generování klíčů pro ElGamalův podpis postupuje A následovně:
1. Vygeneruje velké prvočíslo p a generátor $\alpha \in \mathbb{Z}_p^*$.
2. Zvolí náhodné číslo a, $1 \leq a \leq p-2$.
3. Spočítá $y = \alpha^a \bmod p$.
4. Veřejný klíč entity A je (p, α, y), její soukromý klíč je a.

Nyní následuje samotný podpisový a ověřovací algoritmus:
1. Entita A podepíše binární zprávu m libovolné délky takto:
   (a) Vybere náhodné tajné číslo k, $1 \leq k \leq p-2$, $GCD(k, p-1) = 1$.
   (b) Spočítá $r = \alpha^k \bmod p$.
   (c) Spočítá $k^{-1} \bmod (p-1)$.
   (d) Spočítá $s = k^{-1}\{h(m) - ar\} \bmod (p-1)$.
   (e) Podpis zprávy m entity A je dvojice (r, s).
2. Entita B ověří tento podpis následovně:
   (a) Obdrží veřejný klíč entity A.
   (b) Ověří, zda $1 \leq r \leq p-1$; pokud ne, podpis zamítne.
   (c) Spočítá $v_1 = y^r r^s \bmod p$.
   (d) Spočítá $h(m)$ a $v_2 = \alpha^{h(m)} \bmod p$.
   (e) Přijme podpis právě tehdy, když $v_1 = v_2$.

Důkaz, že ověření podpisu opravdu funguje, je následující:

Pokud byl podpis vygenerován entitou A, pak $s \equiv k^{-1}\{h(m) - ar\} \pmod{p-1}$. Vynásobením obou stran rovnosti číslem k dostaneme $ks \equiv h(m) - ar \pmod{p-1}$, čili $h(m) \equiv ar + ks \pmod{p-1}$. Z toho vyplývá $\alpha^{h(m)} \equiv \alpha^{ar+ks} \equiv (\alpha^a)^r r^s \bmod p$. Takže $v_1 = v_2$, neboť $\alpha^{h(m)} = v_2$ a $(\alpha^a)^r r^s = y^r r^s = v_1$.

1.7 Hašovací funkce a MAC

Zdrojem informací pro tuto sekci jsou knihy [1] a [2].

Hašovací funkce (*Hash function*) je funkce zobrazující binární řetězce libovolné délky na binární řetězce nějaké pevné délky, tyto obrazy se nazývají **haše, otisky** (*hash-values*).

Hašovací funkce by měla splňovat následující požadavky na bezpečnost:

1. **Jednosměrnost** (*pre-image resistance*): pro daný obraz y je výpočetně neproveditelné nalézt vzor x' takový, že $h(x') = y$.

2. **Odolnost vůči kolizi prvního řádu** (*collision resistance*, též *strong collision resistance*): je výpočetně neproveditelné nalézt dva vzory x a x', které mají stejný obraz $h(x) = h(x')$.

3. **Odolnost vůči kolizi druhého řádu** (*2nd pre-image resistance*, též *weak collision resistance*): pro daný vzor x je výpočetně neproveditelné nalézt druhý vzor x' (*2nd pre-image*), $x \neq x'$, který by měl stejný obraz $h(x) = h(x')$.

Hašovací funkce se využívají např. při digitálních podpisech, kdy se podepisuje

haš dlouhé zprávy a straně, která podpis ověřuje, se pošle zpráva spolu s podepsanou haší. Ověřující strana poté vytvoří haš obdržené zprávy a zkontroluje, zda obdržený podpis je platný pro danou haš. V tomto případě je hašovací funkce veřejně dostupná a slouží pouze k urychlení procesu podepisování a ověřování.

Další využití hašovací funkce je pro zajištění integrity dat. Haš daného vstupu se spočítá v nějakou dobu (např. v počátku komunikace). Pro ověření, zda vstupní data nebyla změněna v průběhu komunikace, jsou později opět zhašována a výsledek je porovnán s původní haší. Taková funkce se nazývá *Modification detection code* (MDC).

Mezi veřejně dostupnými hašovacími funkcemi existují také hašovací funkce vyžadující tajný klíč (*keyed hash functions*). Tyto funkce zaručují jak integritu dat, tak autentizaci dat. Jsou to tzv. *Message authentication codes* (MACs).

Message Authentication Code (MAC) je systém funkcí h_k s parametrem k, k je tajný klíč, který má následující vlastnosti:

1. jednoduchost výpočtu - pro známou funkci h_k, daný klíč k a vstup x je jednoduché spočítat hodnotu $h_k(x)$. Tato hodnota se nazývá **MAC** (*MAC-value*) daného vstupu.

2. komprese - funkce h_k zobrazuje vstup x dané délky na výstup $h_k(x)$ pevné délky n bitů.

3. výpočetní odolnost (*computation-resistance*) - (je-li znám popis systému funkcí h) pro každou pevně stanovenou přípustnou hodnotu k, neznámou útočníkovi, a je-li dáno nula či více dvojic textů a MACů $(x_i, h_k(x_i))$, pak je výpočetně neproveditelné vypočítat dvojici $(x, h_k(x))$ pro nějakou novou hodnotu $x \neq x_i$.

1.8 Certifikát s veřejným klíčem

Podkladem pro tuto sekci byla kniha [1].

Certifikát s veřejným klíčem (*Public-key certificate*) slouží k distribuci veřejných klíčů, hlavně ale k zajištění jejich integrity.

Skládá se z **datové části** (*data part*) a z **podpisové části** (*signature part*). Datová část obsahuje jméno entity a veřejný klíč přílušný dané entitě, případně další doplňující data, jako adresa, doba platnosti klíče apod. Podpisová část zahrnuje podpis datové části, vytvořený důvěryhodnou třetí stranou (*trusted third party*, TTP).

Pokud by chtěla entita B ověřit autenticitu veřejného klíče entity A, musí mít B autentickou kopii veřejné ověřovací funkce podpisu TTP. Pak může postupo-

vat takto:

1. Obdrží certifikát s veřejným klíčem entity A (přes nezabezpečený kanál) z centrální databáze certifikátů, přímo od A, nebo jinak.

2. Použije ověřovací funkci třetí důvěryhodné strany pro ověření jejího podpisu v certifikátu entity A.

3. Pokud ověření proběhne v pořádku, přijme veřejný klíč z certifikátu jako autentický veřejný klíč entity A. V opačném případě pokládá veřejný klíč za neplatný.

Předtím, než TTP vytvoří certifikát pro A, musí samozřejmě ověřit její identitu a skutečnost, že veřejný klíč, který má být certifikovaný, opravdu patří A.

Kapitola 2

Dohoda na klíči založená na symetrických technikách

Systém pro distribuci klíčů (KDS) je metoda, při které během inicializační fáze důvěryhodný server generuje a distribuuje tajné hodnoty dat (*pieces*) uživatelům tak, že každá dvojice uživatelů může vypočítat sdílený klíč neznámý všem ostatním uživatelům (vyjma serveru).

Pro fixní párové klíče je KDS schématem pro predistribuci klíče. Triviální příklad KDS je následující: důvěryhodný server vybere navzájem různé klíče pro každou dvojici z n uživatelů (klíčů je tedy $\binom{n}{2}$) a nějakým bezpečným způsobem vždy příslušných $n-1$ klíčů předá jmenovitě každému z uživatelů. Toto zajišťuje bezpodmínečnou (perfektní) bezpečnost - vnější útočník může nanejvýš klíč uhodnout.

Pro posuzování odolnosti KDS vůči větší skupině útočníků je potřeba definovat následující termín.

KDS je j-**bezpečný**, pokud splňuje následující: Je-li dána dvojice uživatelů a jakákoliv skupina j či méně uživatelů (disjunktní s touto dvojicí) dá dohromady své $pieces$, pak se tato skupina nemůže dostat dále při výpočtu klíče sdíleného danými dvěma uživateli než entita, která by hádala klíč bez jakéhokoliv *piece*.

Čili j-**bezpečný** KDS je bezpodmínečně bezpečný proti skupině uživatelů velikosti j a menší.

Příkladem takového triviálního KDS je Blomův protokol pro predistribuci klíče, který je založen na následujícím faktu: V každém j-bezpečném KDS poskytujícím m-bitové párové klíče pro danou relaci, musí být tajná data každého z uživatelů alespoň $m \cdot (j+1)$-bitová.

2.1 Blomův protokol pro predistribuci klíče

Stavebním kamenem tohoto protokolu jsou **lineární samoopravné kódy**, které slouží k detekci a opravování chyb, vzniklých při přenosu dat, konkrétně (n, k) **MDS kódy** (*maximum distance separable codes*).

Lineární kód **délky** n a **dimenze** k je lineární podprostor C dimenze k vektorového prostoru $\mathbb{F}_q^n$, kde $\mathbb{F}_q$ je konečné těleso s q prvky. Takový kód se nazývá q**-ární**.

Lineární kód jakožto podprostor je často reprezentován svou bází. Kódová slova báze jsou uspořádána do řádků **generující matice** G kódu C. Dimenze k lineárního kódu C znamená, že C má ve své bázi k kódových slov, a tedy jeho generující matice má k řádků.

U lineárního kódu je dále definovaná **vzdálenost** d jako Hammingova vzdálenost mezi nějakým daným slovem $c_0 \in C$ a ostatními slovy $c \in C$. Vzdálenost kódu C je nezývislá na výběru $c_0 \in C$, neboť platí

$$min_{c \in C, c \neq c_0} d(c, c_o) = min_{c \in C, c \neq c_0} d(c - c_o, 0) = min_{c \in C, c \neq 0} d(c, 0)$$

(zde vycházím z faktu, že $c - c_0$ je opět slovo kódu C).

Lineární kód délky n, dimenze k a vzdálenosti d se označuje jako (n, k, d) kód.

Podle Singeltonova odhadu (více v [4]) pro každý (n, k, d) kód platí nerovnost $d \leq n - k + 1$. MDS kódy mají největší možnou vzdálenost pro pevné n a k, tedy platí rovnost $d = n - k + 1$ (odtud *maximum distance separable*).

Značení u tohoto protokolu je stejné jako v předchozím textu. Nechť $G = [I_k A]$, kde I_k je jednotková matice.

Každá ze stran potřebuje pouze index i, $1 \leq i \leq n$, který jednoznačně určuje stranu, se kterou bude tvořit sdílený klíč. Před začátkem běhu protokolu obdrží každý z n uživatelů veřejná data a tajný vektor – počáteční klíčový materiál, základní klíč (*base key*), ze kterého je po proběhnutí protokolu schopna každá dvojice uživatelů U_i, U_j vypočítat tajný m-bitový párový klíč $K_{i,j}$ – derivovaný klíč (*derived key*). Z počátečního klíčového materiálu každého z uživatelů lze spočítat více derivovaných klíčů (jeden pro dvojici s každým z ostatních uživatelů).

Protokol:

---

1. Generující matice G typu $k \times n$ MDS kódu nad $\mathbb{F}_q$ je zveřejněna všem n stranám.
2. Důvěryhodná strana T vytvoří tajnou symetrickou matici D typu $k \times k$

nad $\mathbb{F}_q$.

3. T předá každému uživateli U_i tajný klíč S_i.
   S_i je definován jako i-tý řádek $(n \times k)$-matice $S = (DG)^T$, o délce $m = k \cdot lg(q)$ bitů, který umožňuje straně U_i spočítat jakýkoliv prvek matice $(DG)^T G$ v řádku i.

4. Uživatelé U_i a U_j spočítají společný tajný klíč $K_{i,j} = K_{j,i}$ délky m bitů takto:
   U_i použije S_i a j-tý sloupec matice G a spočítá prvek (i,j) ze symetrické $(n \times n)$ matice $K = (DG)^T G$. Podobně U_j za použití S_j a i-tého sloupce matice G spočítá prvek (j, i) ,který je roven prvku (i, j), neboť matice K je symetrická.

Kapitola 3

Transport klíče založený na symetrickém šifrování

V této kapitole jsou protokoly rozděleny do dvou základních skupin - jedna skupina zahrnuje protokoly, kde při transportu klíče důvěryhodný server nehraje roli, v druhé skupině jsou protokoly, které důvěryhodný server vyžadují.

V následujících třech sekcích se zmíním o protokolech, které důvěryhodný server nevyužívají. Používají techniky obnovy klíče, tzv. *Point-to-Point key update* (PTP), které budou vysvětleny níže, a techniku, kdy není potřeba *a priori* sdíleného klíče. Technika PTP je založena buď na symetrickém šifrování nebo na použití jednosměrné funkce.

Poslední tři sekce této kapitoly se zabývají protokoly založenými na používání důvěryhodného serveru, a to zejména KDC. Jsou to Kerberos a s ním související protokoly.

3.1 Obnova klíče PTP založená na symetrickém šifrování

V případě využití symetrického šifrování dvě strany A a B sdílejí *a priori* dlouhodobý symetrický klíč K, který je opakovaně použit pro vytvoření vždy nového kíče W pro danou relaci (*session key*). Značení použité v následujícím textu je takové: t_A je časové razítko, r_A je náhodné číslo a n_A je pořadové číslo, to vše generováno stranou A. E je symetrická šifra. Nepovinné parametry jsou označeny symbolem $*$.

Nyní popíši dvě základní techniky transportu klíče PTP s využitím symetrické šifry:

1. jednoprůchodový transport klíče:

   $A \rightarrow B : E_K(r_A, t_A^*, B^*)$,

   klíč pro danou relaci je $W = r_A$. Obě strany mají zaručenu implicitní autentizaci klíče. Časové razítko (nebo může být použito pořadové číslo) zaručí straně B aktuálnost zprávy. Identifikátor B má zabránit *replay* útoku – odeslání zprávy zpět straně A neidentifikovanou stranou.

   Pokud se chtějí obě strany na klíči podílet, může B poslat A analogickou zprávu s klíčem pro danou relaci, spočítaným jako $f(r_A, r_B)$, kde f by měla být jednosměrná funkce, která zabrání tomu, aby některá ze stran nebo útočník, který zná r_A nebo r_B, mohli ovlivnit hodnotu výsledného klíče.

2. transport klíče s *challenge-response*:

   $A \leftarrow B : n_B$

   $A \rightarrow B : E_K(r_A, n_B, B^*)$

   klíčem pro danou relaci je opět $W = r_A$. Pokud je vyžadováno, aby byl klíč W funkcí vstupních parametrů obou stran A a B, pak komunikace proběhne takto:

   $A \leftarrow B : n_B$

   $A \rightarrow B : E_K(r_A, n_A, n_B, B^*)$

   $A \leftarrow B : E_K(r_B, n_B, n_A, A^*)$

   kde r_A, r_B jsou náhodná čísla sloužící jako klíčový materiál a n_A, n_B jsou *nonce* pro zaručení aktuálnosti.

3.2 Obnova klíče PTP používající jednosměrnou funkci

Případem, kdy PTP využívá jednosměrnou funkci, je protokol AKEP2 (*Authenticated Key Exchange Protocol 2*).

3.2.1 AKEP2

V tomto protokolu si strany A a B vymění tři zprávy a odvodí klíč pro danou relaci W, přičemž je zajištěna vzájemná autentizace entit a implicitní autentizace klíče W.

Na počátku sdílejí A a B dlouhodobé navzájem různé, nikoliv nutně nezávislé, symetrické klíče K, K'.

Funkce h_K je MAC sloužící k autentizaci entit. $h'_{K'}$ je pseudonáhodná permutace nebo jednosměrná funkce, určená k odvození klíče.

Dále je definována hodnota $T = (B, A, r_A, r_B)$. Značení je stejné jako v předchozí sekci.

Protokol:

A: Zvolí náhodné číslo r_A.

1. $A \rightarrow B : r_A$

B: Zvolí náhodné číslo r_B, spočítá $h_K(T)$ s použitím funkce h a klíče K.

2. $A \leftarrow B : T, h_K(T)$

A: Zkontroluje identifikátory v T, dále zda obdržené r_A souhlasí s oním v 1. zprávě a nakonec ověří MAC.

3. $A \rightarrow B : (A, r_B), h_K(A, r_B)$

B: Ověří MAC a zkontroluje, zda obdržené r_B souhlasí s tím, které poslal ve 2. zprávě.
A a B spočítají klíč pro danou relaci $W = h'_{K'}(r_B)$.

3.3 Transport klíče bez předem sdíleného klíče

Vedle technik uvedených v předchozích oddílech patří mezi protokoly, které nevyužívají důvěryhodný server, také Shamirův protokol bez klíče, který, jak je zřejmé již z názvu, nevyžaduje sdílený ani veřejný klíč při dohodě na klíči skrze otevřený kanál a používá pouze symetrické techniky. Každá strana má pouze svůj lokální symetrický klíč.

3.3.1 Shamirův protokol bez klíče

Tento protokol provádí pouze transport klíče, nikoli autentizaci entit či klíče. V podstatě je podobný Diffie-Hellmanově protokolu (viz 5.1), ve kterém si strany narozdíl od tohoto vymění pouze dvě zprávy.

Nejprve se vybere a zveřejní pro společné použití prvočíslo p tak, aby byl problém diskrétního logaritmu modulo p výpočetně nezvládnutelný. Strana A vybere tajné náhodné číslo a, strana B vybere tajné náhodné b, $1 \leq a, b \leq p - 2$,

nesoudělná s $p-1$. Strana A spočítá $a^{-1} \bmod p-1$, B spočítá $b^{-1} \bmod p-1$.

Protokol:

A: Vybere náhodný klíč K, $1 \leq K \leq p-1$ a spočítá $K^a \bmod p$.

1. $A \rightarrow B: K^a \bmod p$

B: Umocní na b obdrženou hodnotu mod p.

2. $A \leftarrow B: (K^a)^b \bmod p$

A: Umocní na $a^{-1} \bmod p-1$ obdrženou hodnotu mod p, čímž dostane $K^b \bmod p$.

3. $A \rightarrow B: K^b \bmod p$

B: Umocní na $b^{-1} \bmod p-1$ obdrženou hodnotu mod p, čímž dostane sdílený klíč $K \bmod p$.

3.4 Kerberos

V protokolu Kerberos strany A (klient) a B (server a verifier) v počátku nesdílejí žádné tajemství, zatímco důvěryhodný server T (*Kerberos authentication server*) sdílí tajemství s oběma (např. uživatelské heslo). B má za úkol ověřit identitu A. T hraje roli KDC (viz výše) a vrací straně A klíč pro danou relaci (*session key*) zašifrovaný pro A a tiket zašifrovaný pro B.

V tomto protokolu se budu držet následujícího značení: E je symetrická šifra. N_A je *nonce* zvolený A, T_A je časové razítko z lokálních hodin A. k je klíč pro danou relaci zvolený T, který má být sdílen stranami A a B.

L je doba platnosti tiketu (*lifetime*), kterou definuje důvěryhodný server T a skládá se z konečného času a případně z počátečního času; tiket je definován takto: $tiket_B := E_{K_{BT}}(k, A, L)$.

L uvnitř tiketu má za úkol eliminovat 1. a 2. zprávu protokolu (tj. strana A nemusí kontaktovat T), pokud by strana A chtěla opětovně použít tiket po skončení omezené doby pro opakovanou identifikaci stranou B. Při každém opětovném použití A vytvoří nový *authenticator* s čerstvým časovým razítkem a stejným klíčem k, který je definován takto: $authenticator := E_k(A, T_A, A^*_{subkey})$, kde symbol $*$ značí volitelný parametr (jeho úloha je vysvětlena níže).

Synchronizované hodiny vždy zajišťuje strana, u níž protokol probíhá (v tomto případě A). Pokud je vstupním klíčem uživatelské heslo, pak bezpečnost závisí pouze na jeho utajení.

Na začátku, jak již bylo zmíněno výše, sdílejí A a T klíč K_{AT}, podobně B a T sdílejí K_{BT}.

Protokol:

A: Vygeneruje N_A.

1. $A \rightarrow T : A, B, N_A$

T: Vygeneruje nový k, definuje L, zašifruje k, N_A, L a B pomocí klíče strany A.
Dále vytvoří tiket zabezpečený klíčem strany B.

2. $A \leftarrow T : tiket_B, E_{K_{AT}}(k, N_A, L, B)$

A: Dešifruje tu část zprávy bez tiketu, získá k, N_A, L a identifikátor B.
Zkontroluje, zda N_A a B souhlasí s hodnotami z 1. zprávy. L si ponechá.
Pomocí k zašifruje vlastní identifikátor A, čerstvé T_A a případně vygenerovaný tajný A_{subkey}, tj. vytvoří $authenticator = E_k(A, T_A, A^*_{subkey})$.

3. $A \rightarrow B : tiket_B, authenticator$

B: Dešifruje tiket, aby získal k a mohl dešifrovat *authenticator*.
Zkontroluje, zda se shoduje identifikátor A v tiketu a v *authenticator*. Dále zkontroluje platnost T_A v *authenticator*. Nakonec zkontroluje, zda lokální čas strany B odpovídá času vymezenému L.
Jestliže je vše v pořádku, autentizace A je považována za úspěšnou.
Pokud je použit A_{subkey}, uloží jej.

Pro případnou vzájemnou autentizaci entit:

4. $A \leftarrow B : E_k(T_A, B^*_{subkey})$

A: Dešifruje 4. zprávu.
Zkontroluje, zda T_A souhlasí s tím ve 3. zprávě.
Jestliže je vše v pořádku, považuje autentizaci B za úspěšnou.

Zde volitelné parametry A_{subkey} a B_{subkey} slouží k předání klíče (jiného než k)

mezi A a B nebo k výpočtu složeného klíče za použití nějaké funkce $f(A_{subkey}, B_{subkey})$.

3.5 Needhamův-Schroederův protokol se sdíleným klíčem

Tento protokol využívá server a je základem protokolů pro transport klíče. Je nezávislý na časovém razítku a provádí jak autentizaci entity, tak autentizaci klíče.

Značení u tohoto protokolu je následující: E je symetrická šifra. N_A je *nonce* zvolený stranou A a N_B je *nonce* zvolený stranou B. Klíč k je klíč pro danou relaci (*session key*) zvolený důvěryhodným serverem T, který má být sdílen stranami A a B po ukončení protokolu.

V úvodu sdílejí A a T klíč K_{AT}, podobně B a T sdílejí K_{BT}.

4. a 5. zpráva zajistí prokázání A straně B (ve smyslu autentizace entity).

Pokud by strana A chtěla opětovně použít k v komunikaci s B, pak si A bezpečně uloží data poslaná ve 3. zprávě spolu s k. Při dalším použití lze vynechat 1. a 2. zprávu. Na konec 3. zprávy je potřeba přidat zašifrovaný *nonce* $E_k(N_A^{'})$ a 4. zprávu nahradit $E_k(N_A^{'} - 1, N_B)$, aby strana A ověřila, že strana B v tuto chvíli opravdu zná hodnotu k (prevence *replay* útoku pomocí starší zprávy č. 4).

Protokol probíhá podobně jako Kerberos, až na ověřování *nonce*.

Protokol:

---

1. $A \rightarrow T : A, B, N_A$
2. $A \leftarrow T : E_{K_{AT}}(N_A, B, k, E_{K_{BT}}(k, A))$
3. $A \rightarrow B : E_{K_{BT}}(k, A)$
4. $A \leftarrow B : E_k(N_B)$
5. $A \rightarrow B : E_k(N_B - 1)$

Slabinou tohoto protokolu je, že nezaručuje aktuálnost klíče k. Proto může útočník kdykoliv později znovu poslat 3. zprávu a poté i spočítat obsah 5. zprávy. U protokolu Kerberos je toto ošetřeno parametrem $lifetime$.

3.6 Otwayův-Reesův protokol

Tento protokol opět využívá server a zajišťuje autentizaci klíče a jeho aktuálnost. Neprovádí však autentizaci entit ani potvrzení přijetí klíče.

Značení v tomto protokolu je následující: E je symetrická šifra. N_A je *nonce* zvolený A, N_B je *nonce* zvolený B. k je klíč pro danou relaci (*session key*) zvolený T, který má být sdílen stranami A a B. M je druhý *nonce* zvolený A, který slouží jako identifikátor transakce.

Na počátku sdílejí A a T klíč K_{AT}, podobně B a T sdílejí K_{BT}.

Protokol:

A: Zašifruje data určená serveru: *nonce* N_A a M, svůj identifikátor a identifikátor strany B, které má server předat klíč.

1. $A \rightarrow B : \ M, A, B, E_{K_{AT}}(N_A, M, A, B)$

B: Vytvoří vlastní *nonce* N_B, analogicky zašifruje data pro server vlastním klíčem a připojí k nim data obdržená od strany A.

2. $B \rightarrow T : \ M, A, B, E_{K_{AT}}(N_A, M, A, B), E_{K_{BT}}(N_B, M, A, B)$

T: Použije nezašifrované identifikátory A a B k rozpoznání klíčů K_{AT}, K_{BT}, poté ověří zda nezašifrovaná sekvence (M, A, B) souhlasí s oběma dešifrovanými sekvencemi
bez N_A, resp. N_B.
Pokud vše souhlasí, zašifruje nový klíč k spolu s odpovídajícími *noncemi* pomocí patřičného klíče každé ze stran.

3. $B \leftarrow T : \ E_{K_{AT}}(N_A, k), E_{K_{BT}}(N_B, k)$

B: Dešifruje druhou část zprávy, zkontroluje, zda N_B odpovídá tomu, které poslal ve 2. zprávě, a pokud ano, přepošle pvní část zprávy straně A.

4. $A \leftarrow B : \ E_{K_{AT}}(N_A, k)$

A: Dešifruje zprávu a zkontroluje, zda N_A odpovídá tomu, které poslal v 1. zprávě. Pokud ano, protokol je úspěšně ukončen.

Po proběhnutí protokolu jsou A a B ujištěni, že k je čerstvý. Věří, že třetí strana T, se kterou sdílí k, je strana vázaná jejich *noncemi* ve 2. zprávě. A ví, že B je aktivní, jelikož z ověření 4. zprávy vyplývá, že B poslal zprávu před velmi krátkou dobou. B však nemá jistotu, že A je aktivní, dokud A nepoužije k v další komu-

nikaci, jelikož B není schopen určit, zda je 1. zpráva aktuální (*fresh*).

Otwayův-Reesův protokol je napadnutelný přepisovacím útokem (*typing attack*). Útok spočívá v podobnosti zašifrovaných částí v 1. a 4. zprávě - obě jsou zašifrovány stejným klíčem K_{AT} a obě začínají polem stejného typu. Útok bude fungovat v případě, kdy bitová délka (M, A, B) bude shodná s délkou očekávaného klíče k. Tedy např. M může být délky 64 bitů, A a B mohou být délky 32 bitů a klíč k může být délky 128 bitů.

Za těchto podmínek bude útok, kde C_B je útočník vydávající se za B, probíhat následovně:

Útočník C_B zachytí zprávu od strany A:

1. $A \rightarrow C_B : \ M, A, B, E_{K_{AT}}(N_A, M, A, B)$

Útočník vrátí zašifrovanou část zprávy straně A, která ji považuje za 4. zprávu protokolu.

4. $C_B \rightarrow A : \ E_{K_{AT}}(N_A, M, A, B)$

A přijme pole (M, A, B) jako sdílený klíč k. Útočník může pokračovat v další komunikaci se stranou A, zabezpečené tímto klíčem, a vydávat se za stranu B.

Předejít tomuto typu útoku lze tak, že se v každé zprávě změní pořadí prvků zprávy, nebo se do zpráv přidá autentizované číslo zprávy či autentizovaný typ ke každé položce pole. To vše samozřejmě zvýší výpočetní náročnost protokolu.

Kapitola 4

Transport klíče založený na šifrování s veřejným klíčem

Zde jedna strana vybere symetrický klíč a předá jej straně druhé, přičemž použije šifru s veřejným klíčem druhé strany. Toto zaručí odesílateli autentizaci klíče. Autentizace klíče příjemci a potvrzení přijetí klíče lze řešit například přídavnou zprávou, digitálním podpisem či použitím symetrické šifry spolu s podpisem.

Autentizace může být zajištěna

- autentizací entity skrze dešifrování pomocí veřejného klíče,
- autentizací dat prostřednictvím digitálního podpisu.

Mezi protokoly, které provádí transport klíče za použití veřejného klíče bez podpisu, se řadí mimo jiné Needham-Schroederův protokol s veřejným klíčem, který zaručuje vzájemnou autentizaci entit.

Aby bylo vedle zabezpečení klíčových materiálů dosaženo autentizace zdroje, lze použít kombinace šifrování s veřejným klíčem a podpisů, a to způsoby popsanými v následujících odstavcích.

Značení v následujícím textu je takovéto: P_B je šifra s veřejným klíčem strany B, k je klíč, t_A je časové razítko, S_A je podpisová funkce a B je identifikátor. Nepovinné parametry jsou označeny symbolem $*$.

1. Zašifrování podepsaného klíče:

   $A \rightarrow B : \ P_B(k, t_A^*, S_A(B, k, t_A^*))$.

   V případě použití podpisu, umožňujícího získání původní zprávy z podpisu (*message recovery*), lze (a je nutné) výše zmíněný postup upravit:

   $A \rightarrow B : \ P_B(S_A(B, k, t_A^*))$.

2. Zašifrování a podpis klíče zvlášť:

   $A \rightarrow B: \ P_B(k, t_A^*), S_A(B, k, t_A^*),$

   pro případ podpisu, ze kterého nelze získat původní zprávu ani žádné informace o otevřeném textu (např. pokud podpisová funkce zahrnuje jednosměrnou hašovací funkci).

3. Podepsání zašifrovaného klíče:

   $A \rightarrow B: \ t_A^*, P_B(A, k), S_A(B, t_A^*), P_B(A, k)),$

   bez přítomnosti parametru A uvnitř podepsaného řetězce by bylo možno extrahovat z podpisu hodnotu $P_B(k)$ a podepsat ji vlastním (útočníkovým) klíčem (tj. znemožnění autentizace). Navíc je potřeba, aby použitá šifra zamezila útočníkovi nahradit $P_B(A, k)$ za $P_B(C, k)$ i přes neznalost k.

   Pro vzájemnou autentizaci entit se navíc používají buď to časová razítka ve dvoucestném protokolu (viz dále - protokol X.509 dvoucestný) nebo metoda *challenge-response* (výzva-odezva) v třícestném protokolu (viz X.509 třícestný), která vypadá následovně:

   r_A a r_B jsou náhodná čísla vygenerovaná stranami A a B (v tomto pořadí).

   $A \rightarrow B: \ r_A$

   $A \leftarrow B: \ r_B, P_A(B, k_1), S_B(r_A, r_B, A, P_A(B, k_1))$

   $A \rightarrow B: \ P_B(A, k_2), S_A(r_B, r_A, B, P_B(A, k_2))$

4. Použití navíc symetrické šifry k šifře s veřejným klíčem a podpisu.

   Jsou to tzv. hybridní protokololy pro transport klíče; hybridní se nazývají proto, že používají jak symetrické tak asymetrické techniky. Příkladem jsou Bellerovy-Yacobiho protokoly - čtyřcestný a dvoucestný, které budou popsány v poslední sekci této kapitoly.

Podpisová schémata, která mohou být použita, jsou zejména RSA podpis, Rabinův podpis a ElGamalův podpis.

4.1 Needham-Schroederův protokol s veřejným klíčem

Tento protokol zaručuje vzájemnou autentizaci entit a vzájemný transport symetrického klíče. Předané klíče mohou sloužit nejen jako *nonce* pro autentizaci entit, ale i jako tajné klíče pro další použití. Kombinací výsledných klíčů lze vypočítat společný klíč, na jehož výpočtu se obě strany podílí.

Značení u tohoto protokolu je následující: $P_A(Y)$ jsou data Y zašifrovaná šifrou s veřejným klíčem strany A; $P_A(Y_1, Y_2)$ značí zašifrování konkatenace Y_1 a Y_2. Klíče k_1, k_2 jsou stranami A a B (v tomto pořadí) vybrané tajné symetrické klíče pro danou relaci (*session keys*).

Předpokládáme, že A a B již mají navzájem své veřejné klíče.

Protokol:

---

1. $A \rightarrow B : \ P_B(k_1, A)$

B: Po dešifrování zprávy získá k_1 a odešle následující zprávu.

2. $A \leftarrow B : \ P_A(k_1, k_2)$

A: Porovná k_1 obdržený ve 2. zprávě a k_1 poslaný v 1. zprávě. Tím, že k_1 nebyl nikdy předtím použit, je provedena autentizace entity (strany B) a zároveň je A ujištěn, že B zná klíč.

3. $A \rightarrow B : \ P_B(k_2)$

B: Porovná k_2 obdržený ve 3. zprávě a poslaný v 2. zprávě. Klíč pro danou relaci (*session key*) může být spočítán jako $f(k_1, k_2)$ použitím nějaké veřejně dostupné nevratné funkce f.

Tento protokol může být modifikován eliminací šifry v poslední zprávě, pokud k_1 a k_2 nemají sloužit jako sdílená tajemství. Nechť r_1 a r_2 jsou náhodná čísla generovaná A a B (v tomto pořadí). Pak zprávy protokolu vypadají následovně:

1′. $A \rightarrow B : \ P_B(k_1, A, r_1)$

2′. $A \leftarrow B : \ P_A(k_2, r_1, r_2)$

3′. $A \rightarrow B : \ r_2$

Needhamův-Schroederův protokol (první zmíněná verze) je napadnutelný Loweovým útokem, neboť protokol nezaručuje straně B, že poslední zpráva pochází od strany A. Útočník C přiměje stranu A k započetí běhu protokolu a poté, vydávajíce se za stranu A započne protokol se stranou B. Útok probíhá takto:

1. $A \rightarrow C : P_C(k_1, A)$

1'. $C_A \rightarrow B : P_B(k_1, A)$

2'. $B \rightarrow C_A : P_A(k_1, k_2)$

2. $C \rightarrow A : P_A(k_1, k_2)$

3. $A \rightarrow C : P_C(k_2)$

3'. $C_A \rightarrow B : P_B(k_2)$

Lowe tedy navrhnul variantu Needhamova-Schroederova protokolu, který má být právě odolný vůči takovému útoku, a to takovou, kde v druhé zprávě je navíc zahrnut identifikátor strany B:

1. $A \rightarrow B : P_B(k_1, A)$

2. $A \leftarrow B : P_A(k_1, k_2, B)$

3. $A \rightarrow B : P_B(k_2)$

4.2 Silně autentizační protokoly X.509

Jak již bylo zmíněno výše, tyto protokoly využívají kromě šifry s veřejným klíčem také techniku podepsání zašifrovaného klíče. Jak dvoucestný tak třícestný protokol X.509 poskytují vzájemnou autentizaci entit. Výraz „silně" zdůrazňuje odlišnost od jednodušších metod založených pouze na heslu.

V protokolech jsou vyměněny dvě zprávy za použití časového razítka či tři zprávy s použitím *challenge-response* s náhodnými čísly.

Oba protokoly zajišťují straně B:

1. identifikaci A a jistotu, že data, která obdržela strana B, byla vytvořena stranou A (autentizace dat) a nebyla později změněna (integrita dat),

2. že data obdržená stranou B byla výslovně určena straně B,

3. že data obdržená stranou B jsou aktuální (*fresh*),

4. vzájemnou bezpečnost předávaného klíče.

   Totéž oba protokoly zaručují i straně A.

Pro tyto protokoly je použito následující značení: $P_A(Y)$ značí zašifrovaná data Y za použití veřejného klíče strany A. Výraz $S_A(Y)$ značí podepsaná data Y za použití soukromého klíče strany A. Výrazy r_A, r_B jsou čísla na jedno použití (pro detekci *replay* útoku nebo situace, kdy by se útočník chtěl vydávat za jednu ze zúčastněných stran). Výraz $cert_A$ je certifikát spojující stranu A a veřejný klíč (nebo pár veřejných klíčů) vhodný pro podpis a ověření podpisu.

Každá ze stran má svůj pár veřejných klíčů pro podpisy a šifrování. Strana A musí získat (a autentizovat) *a priori* veřejný klíč pro šifrování strany B.

Nechť $D_A = (t_A, r_A, B, data_1^*, P_B(k_1)^*), D_B = (t_B, r_B, A, r_A, data_2^*, P_A(k_2)^*)$. Parametry označené symbolem $*$ jsou volitelné. Výrazy t_A a t_B jsou časová razítka stran A a B (v tomto pořadí).

Protokol X.509 - dvoucestný:

A: Obdrží t_A, vygeneruje r_A, případně obdrží symetrický klíč k_1. $data_1$ je volitelný údaj určený k autentizaci dat.

1. $A \rightarrow B : cert_A, D_A, S_A(D_A)$

B: Ověří autenticitu $cert_A$, extrahuje podpisový veřejný klíč strany A a ověří její podpis v bloku D_A. Dále zkontroluje identifikátor příjemce B, zda je platné časové razítko a nakonec je-li r_A použito poprvé. Pokud je vše při kontrole v pořádku, pak B dešifruje k_1 za použití svého soukromého klíče a tento aktuálně sdílený klíč uloží. Jednostranná autentizace (autentizace A) tímto končí.

2. $A \leftarrow B : cert_B, D_B, S_B(D_B)$

Zde se vše analogicky opakuje. Pokud je opět při kontrole vše v pořádku, A uloží klíč k_2 pro následné použití.
A a B sdílejí vzájemná tajemství k_1 a k_2.

Standard X.509 převzal schéma s veřejným klíčem RSA, které umožňuje použít stejný pár klíčů pro šifrovací i pro podpisový algoritmus. Je však lepší použít vždy jiný pár klíčů pro šifrování a jiný pro podpis.

Nevýhodou tohoto protokolu je, že jelikož v něm není zahrnut identifikátor (např. A) uvnitř funkce P_B v sekvenci D_A, není možno zaručit, že podepisující strana opravdu zná nebo je strůjcem klíče k_1.

Protokol X.509 - třícestný:

Liší se od dvoucestného v následujícím:

1. Časová razítka se nastaví na nulu a nemusí být následně kontrolována. 2. Po obdržení 2. zprávy A zkontroluje, zda obdržené r_A odpovídá tomu z 1. zprávy. 3. Dále je poslána 3. zpráva:

$A \rightarrow B : (r_B, B), S_A(r_B, B)$

4. Po obdržení 3. zprávy B ověří, zda obsah podpisu souhlasí s nepodepsanou sekvencí, jestli je identifikátor B správný a zda r_B v této zprávě souhlasí s tím ve 2. zprávě.

4.3 Bellerův-Yacobiho protokol pro transport klíče

Tento protokol pro transport klíče zajišťuje vzájemnou autentizaci entit a vzájemnou explicitní autentizaci klíče a byl navržen za účelem minimalizovat výpočetní požadavky na jednu z komunikujících stran (slabší stranu, zde A). Další vlastností tohoto protokolu je, že identita strany A zůstává v utajení (před útočníky).

Strana A se autentizuje straně B podepsáním náhodné výzvy (*challenge*) m. Strana B se autentizuje straně A prokázáním znalosti klíče K, který může získat jen on sám.

Nyní popíši Bellerův-Yacobiho čtyřcestný protokol pro transport klíče, který zde pro jednoduchost používá šifrovací algoritmus RSA s veřejným exponentem 3. V praxi je však doporučeno používat Rabinův algoritmus pro šifru s veřejným klíčem, neboť je mnohem efektivnější.

Značení u tohoto protokolu je následující: $E_K(y)$ je y zašifrované symetrickou šifrou E s použitím klíče K. $P_X(y)$ je výsledek aplikování funkce s veřejným klíčem strany X na y. $S_X(y)$ je výsledek aplikování funkce se soukromým klíčem strany X na y.

I_X značí identifikační řetězec strany X (např. jméno a adresa). $h(y)$ je haš y, použitá v souvislosti s podpisovým schématem. Pokud je $y = y_1, ..., y_n$, pak

vstup je konkatenace dílčích hodnot.

Nejprve je nastaven systém: Vhodné prvočíslo n_S a generátor α multiplikativní grupy modulo n_S jsou pevně stanoveny jako parametry ElGamalovy šifry. Důvěryhodný server T vybere vhodná prvočísla p a q, $pq = n_T$ je modulus pro RSA podpis. Dále pro veřejný exponent $e_T = 3$ vypočítá soukromý klíč d_T, splňující $e_T d_T \equiv 1 \bmod (p-1)(q-1)$.

Tyto systémové parametry jsou rozdány stranám A a B takto: Každá strana obdrží autentickou kopii veřejného klíče důvěryhodného serveru T a parametry $n_T, (n_S, \alpha)$. T předá jmenovitě každé straně X identifikační řetězec I_X.

Dále je nastaven terminál: Každá strana hrající roli A (terminálu) vybere náhodné číslo a, $1 \leq a \leq n_S - 2$, a spočítá veřejný klíč pro ElGamalův podpis $u_A = \alpha^a \bmod n_S$.

Strana A si ponechá tajný soukromý klíč a a předá veřejný klíč u_A důvěryhodného serveru T jinou cestou, než probíhala předešlá komunikace (např. osobně).

Důvěryhodný server T vytvoří a vrátí straně A certifikát s veřejným klíčem $cert_A = (I_A, u_A, G_A)$. Výraz G_A je RSA podpis dat (I_A, u_A) důvěryhodným serverem T, čili $G_A = S_T(I_A, u_A) = (h(I_A, u_A))^{d_T} \bmod n_T$.

Jako další je nastaven server: Každá strana hrající roli B (serveru) vytvoří soukromý klíč a k němu patřičný veřejný klíč pro šifru RSA s veřejným exponentem $e_B = 3$. B vybere modulus n_B pro šifru s veřejným klíčem a spočítá odpovídající soukromý klíč d_B.

B předá n_B důvěryhodnému serveru T opět jinou cestou (např. osobně).

T vytvoří a vrátí B certifikát s veřejným klíčem $cert_B = (I_B, n_B, G_B)$, RSA podpis $G_B = S_T(I_B, n_B) = (h((I_B, n_B))^{d_T} \bmod n_T$.

Protokol:

A: Vybere náhodné x, $1 \leq x \leq n_S - 2$, a spočítá tyto tři hodnoty:
$v = \alpha^x \bmod n_S$
$x^{-1} \bmod (n_S - 1)$
$av \bmod (n_S - 1)$
Pro bezpečnost ElGamalova podpisu je nutné, aby x bylo pro každý podpis nově zvolené a nesoudělné s $n_S - 1$, což zaručí existenci x^{-1}.

1. $A \leftarrow B : \ cert_B = (I_B, n_B, G_B)$

A: Zkontroluje autenticitu n_B tak, že ověří, zda $h(I_B, n_B) = G_B^3 \bmod n_T$. Dále vybere náhodný klíč K, $1 < K < n_B - 1$, a zašifruje jej: $Y = P_B(K)$.

2. $A \rightarrow B: \ P_B(K) = K^3 \bmod n_B$

B: Získá $K = S_B(Y) = Y^{d_B} \bmod n_B$. Dále vybere náhodné m (*challenge*), doplní ho t nulami (*least significant bits*) a symetricky zašifruje pomocí K.

3. $A \leftarrow B: \ E_K(m, \{0\}^t)$

A: Dešifruje obdrženou zprávu a poté zkontroluje, zda na konci obsahuje t nul. Pokud ano, ví, že původcem zprávy je B a že B zná klíč K.
Nyní přidá dešifrované m k identifikátoru I_B strany, jejíž veřejný klíč použil pro sdílení K ve 2. zprávě, čili vytvoří $M = (m, I_B)$. Pak spočítá w takové, že $w \equiv (M - av) \cdot x^{-1} \bmod (n_S - 1)$.
Výraz (v, w) je ElGamalův podpis zprávy M.

4. $A \rightarrow B: \ E_K((v, w), cert_A)$

B: Dešifruje obdrženou zprávu a ověří autenticitu u_A tak, že zkontroluje, zda $h(I_A, u_A) = G_A^3 \bmod n_T$.
Dále vytvoří $M = (m, I_B)$ z hodnoty m, kterou poslal ve 3. zprávě, a z vlastního identifikátoru a ověří podpis strany A tak, že zkontroluje, zda $\alpha^M \equiv u_A^v \cdot v^w \bmod n_S$.
Pokud vše souhlasí, přijme B stranu A jako stranu odpovídající identifikátoru I_A a jako původce klíče K.

Kapitola 5

Dohoda na klíči založená na asymetrických technikách

Základní technikou provádějící neautentizovanou dohodu na klíči je Diffie-Hellmanova (DH) dohoda na klíči. Tento protokol a protokoly s ním související jsou založeny na problému výpočtu diskrétního logaritmu (DLP), implicitně certifikovaných veřejných klíčích a jejich použití v DH protokolech.

5.1 DH a s ním související protokoly pro dohodu na klíči

Jednou z variant DH je ElGamalova dohoda na klíči, která provádí jednostrannou autentizaci klíče zamýšleného příjemce odesílateli, kde je veřejný klíč příjemce znám odesílateli *a priori*. Tento protokol je zjednodušený DH.

Další variantou Diffie-Hellmanova protokolu jsou MTI dvoucestné protokoly pro dohodu na klíči. MTI/A0 nevyžaduje podpis a produkuje časově variabilní klíč pro danou relaci se vzájemnou autentizací klíče odolný vůči pasivním útokům. Neprovádí však autentizaci entit ani ověření klíče. Více o protokolu MTI/A0 lze nalézt na str. 40 a v [1], str. 518.

Poslední zmíněnou variantou DH je třícestný protokol Station-to-Station (STS), který umožňuje vytvoření sdíleného tajemství mezi dvěma stranami se vzájemnou autentizací entit a vzájemnou explicitní autentizací klíče. Protokol také podporuje anonymitu, tj. entity A a B jsou chráněny proti odposlechu, a využívá digitální podpis, např. RSA.

5.1.1 Diffie-Hellmanova dohoda na klíči

Tato technika byla prvním praktickým řešením problému distribuce klíče, kdy se dvě strany nikdy dříve nesetkaly, nikdy nesdílely žádný klíčový materiál

a chtěly výměnou zpráv skrze otevřený kanál vytvořit tajný klíč, který by sdílely. Bezpečnost spočívala v neprolomitelnosti Diffie-Hellmanova problému a s ním souvisejícího problému diskrétního logaritmu.

Základní verze poskytuje ochranu výsledného klíče před pasivním útočníkem, nikoli však před aktivním útočníkem schopným zachytit zprávu, modifikovat ji či vložit zprávu do komunikace. Žádná ze stran nemá záruku autentizace entit ani autentizace klíče.

Na začátku protokolu je dáno a zveřejněno vhodné prvočíslo p, dále generátor $\alpha \in \mathbb{Z}_p^*$ ($2 \leq \alpha \leq p-2$).

Protokol:

A: Vybere náhodné tajné číslo x, $1 \leq x \leq p-2$ a pošle následující zprávu:

1. $A \rightarrow B : \alpha^x \bmod p$

B: Obdrží α^x a spočítá klíč ke sdílení $K = (\alpha^x)^y \bmod p$. Dále vybere náhodné tajné y, $1 \leq y \leq p-2$, a pošle následující zprávu:

2. $A \leftarrow B : \alpha^y \bmod p$

A: obdrží α^y a spočítá klíč $K = (\alpha^y)^x \bmod p$.

Varianta tohoto protokolu s fixními výrazy α^x a α^y provádí vzájemnou autentizaci. Exponenciální výrazy α^x a α^y slouží jako dlouhodobé veřejné klíče patřičných stran a distribuují se za použití podepsaných certifikátů, tudíž se pevně stanoví dlouhodobý sdílený klíč $K = \alpha^{xy}$ pro tuto dvojici uživatelů. Pokud jsou takové certifikáty dostupné *a priori*, pak není potřeba posílat zprávy (nulacestný protokol). Nevýhodou takového klíče K je jeho časová neměnnost.

Řešením může být dohoda na klíči MTI/A0, kde je kromě prvočísla p a generátoru $\alpha \in \mathbb{Z}_p^*$ zvoleno stranou A jako dlouhodobý soukromý klíč celé číslo a, $1 \leq a \leq p-2$, a spočten dlouhodobý veřejný klíč $z_A = \alpha^a \bmod p$. Strana B má analogicky klíče b, z_B. Zprávy, které si tyto dvě strany pošlou, se od výše zmíněných neliší, pouze A spočte klíč $k = (\alpha^y)^a z_B^x \bmod p$ a B spočte $k = (\alpha^x)^b z_A^y \bmod p$; obě strany nyní sdílejí klíč $k = \alpha^{(bx+ay)} \bmod p$.

Protokol může být uskutečněn v jakékoli grupě, kde je problém diskrétního logaritmu obtížně řešitelný a umocňování dostatečně efektivní. Nejčastěji se používá multiplikativní grupa $\mathbb{Z}_p^*$ nebo $\mathbb{F}_{2^m}^*$, případně grupa bodů definovaných eliptickou křivkou nad konečným tělesem.

Použití nevhodného soukromého exponentu x znamená restrikci výsledného klíče. Nejhorší možný případ volby soukromého exponentu v $\mathbb{Z}_p$ by byl 0, jelikož dává $\alpha^0 = 1$ a výsledný klíč by byl $K = 1$ nezávisle na volbě y, proto se v protokolu předpokládá $x \geq 1$. Takže některé varianty DH protokolu nejsou odolné vůči následujícímu aktivnímu útoku:

Nechť α generuje $\mathbb{Z}_p^*$, kde $p = Rq + 1$, q je prvočíslo a nechť např. $R = 2$. Pak prvek $\beta = \alpha^q = \alpha^{(p-1)/R}$ je řádu R, speciálně pro $R = 2$ je $\beta = -1$.

Takže když si A a B vyměňují neautentizované krátkodobé exponenciální výrazy α^x a α^y, útočník je může nahradit hodnotami $(\alpha^x)^q$ a $(\alpha^y)^q$, čímž si vynutí sdílený klíč $K = \alpha^{xyq} = \beta^{xy}$, který může nabývat pouze R hodnot (speciálně pro $R = 2$ může klíč K nabývat hodnot ± 1).

Více přímočarý útok spočívá v jednoduchém nahrazení jednoho z exponenciálních výrazů α^x, α^y za hodnoty $+1$ nebo $p - 1 = -1$.

Tomuto typu útoků se předejde např. použitím digitálního podpisu vzájemně vyměněných exponenciálních výrazů α^x a α^y.

5.1.2 Protokol Station-to-Station

E je algoritmus pro symetrickou šifru. $S_A(m)$ značí podpis m stranou A, definovaný jako $S_A(m) = (H(m))^{d_A} \bmod n_A$, tj. metoda RSA předcházená příslušnou jednosměrnou hašovací funkcí H, $H(m) < n_A$.

Nejprve je vybráno a zveřejněno vhodné prvočíslo p a generátor $\alpha \in \mathbb{Z}_p^*$, $2 \leq \alpha \leq p - 2$. Pro zvýšenou bezpečnost může mít každá strana své vlastní parametry jako součást svých veřejných klíčů.

Uživatel A zvolí veřejný klíč (e_A, n_A) a soukromý klíč d_A pro RSA podpis, analogicky B zvolí (e_B, n_B), d_B. Předpokládáme, že každá ze stran má přístup ke kopiím veřejných klíčů ostatních stran. Pokud ne, pak musí být zahrnuty ve zprávách (2) a (3) certifikáty.

Protokol:

A: Vygeneruje tajné náhodné číslo x, $1 \leq x \leq p - 2$, a pošle 1. zprávu.

1. $A \rightarrow B$: $\alpha^x \bmod p$

B: Stejným způsobem vybere číslo y a vypočítá sdílený klíč $k = (\alpha^x)^y \bmod p$. Dále podepíše konkatenaci obou exponenciálních výrazů α^y a α^x v tomto pořadí, to zašifruje pomocí vypočítaného klíče a pošle 2. zprávu.

2. $A \leftarrow B : \alpha^y \bmod p, E_k(S_B(\alpha^y, \alpha^x))$

A: Spočítá sdílený klíč $k = (\alpha^y)^x \bmod p$, dešifruje zašifrovaná data a použije veřejný klíč strany B k ověření, že podepsaná hodnota je haš obdrženého nezašifrovaného exponenciálního výrazu α^y (taktéž obdrženého v této zprávě) a exponenciálního výrazu poslaného v 1. zprávě. Po úspěšném ověření považuje toto k za opravdu sdílené se stranou B a pošle analogickou 3. zprávu.

3. $A \rightarrow B : E_k(S_A(\alpha^x, \alpha^y))$

B: Podobně dešifruje obdrženou zprávu a ověří podpis. Pokud úspěšně, pak přijme k za opravdu sdílený se stranou A.

Šifrování pomocí klíče k zaručí vzájemné potvrzení přijetí klíče a připustí závěr, že strana, která tento klíč zná, je zároveň stranou, která podepsala ony exponenciální výrazy. Nejlepší případ použití tohoto protokolu nastane, pokud všechny další zprávy budou také zašifrovány pomocí klíče k. V jiném případě se potvrzení přijetí klíče dosáhne např. použitím MAC ve 2. a 3. zprávě - pro $s = S_A(\alpha^x, \alpha^y)$, $A \rightarrow B : (s, MAC_k(s))$. Jinou možností je zahrnutí jednosměrné hašovací funkce uvnitř podepsané zprávy - $A \rightarrow B : S_A(\alpha^x, \alpha^y, h(k))$, kde $h(k)$ může být nahrazeno k v případě, že podpis sám zahrnuje příslušnou jednosměrnou hašovací funkci.

5.2 Implicitně certifikované veřejné klíče

Jiný způsob distribuce klíče, narozdíl od systémů používajících certifikáty s veřejným klíčem, je použití **implicitně certifikovaných veřejných klíčů** (*implicitly-certified public keys*). Takové klíče, které mouhou být rekonstruovány z veřejných dat, mohou být použity v protokolech pro dohodu na klíči, které vyžadují certifikované DH veřejné klíče (např. MTI/A0) jako alternativu přenosu těchto klíčů pomocí certifikátů s veřejným klíčem.

Implicitně certifikované klíče jsou veřejné klíče, které již existují, ale musí být rekonstruovány z nějakých veřejně dostupných dat, jak již bylo zmíněno výše. Taková data zahrnují veřejná data související s důvěryhodnou stranou T, dále zahrnují identifikační údaje entity (např. jméno, adresu apod.) a doplňující veřejná data pro uživatele - **veřejná data pro rekonstrukci** (*reconstruction public data*). Integrita rekonstruovaného veřejného klíče se nedá přímo ověřit, avšak správný veřejný klíč se dá získat pouze z autentických veřejných dat uživatele.

Při změně uživatelovy identity či veřejných dat pro rekonstrukci (*reconstruc-*

tion public data) se získá „padělaný" veřejný klíč, který sice způsobí zamítnutí služeb (*denial of services*), nebude však z kryptografického hlediska riskantní. Kryptografická transformace prostě selže (selže ověření podpisu, šifra s veřejným klíčem vyprodukuje nedešifrovatelný text a dohoda na klíči skončí tak, že uživatelé vytvoří navzájem různé klíče).

Pokud útočník nezná veřejná data důvěryhodné strany T, pak je pro něj výpočetně neproveditelné spočítat soukromý klíč odpovídající veřejnému klíči uživatele, nebo vytvořit odpovídající identifikaci uživatele a veřejná data pro rekonstrukci (*reconstruction public data*), podle kterých má být soukromý klíč vypočítán. Rekonstruované veřejné klíče jsou tedy implicitně autentizované samotnou konstrukcí (odtud pojem „implicitně certifikovaný").

5.2.1 Güntherův mechanismus s implicitně certifikovanými veřejnými klíči

Důvěryhodný server T vybere vhodné pevně stanovené veřejné prvočíslo p a generátor $\alpha \in \mathbb{Z}_p^*$, dále vybere jako svůj soukromý klíč náhodné celé t, $1 \leq t \leq p-2$, $GCD(t, p-1) = 1$, a zveřejní svůj veřejný klíč $u = \alpha^t \bmod p$ spolu s α a p.

T určí pro každou stranu A unikátní jméno či identifikační řetězec I_A (např. jméno a adresu) a náhodné celé číslo k_A, $GCD(k_A, p-1) = 1$. Pak spočítá $P_A = \alpha^{k_A} \bmod p$, P_A jsou veřejná data pro rekonstrukci strany A. Podmínka GCD zajišťuje, že P_A je sám generátor.

Použitím vhodné hašovací funkce h, T vyřeší následující rovnici pro a:

$$h(I_A) \equiv t \cdot P_A + k_A \cdot a \pmod{p-1}.$$

T bezpečně předá straně A dvojici $(r, s) = (P_A, a)$, což je ElGamalův podpis I_A; a je tajný klíč strany A pro DH dohodu na klíči.

Jakákoli další strana nyní může rekonstruovat (DH) veřejný klíč strany A, totiž $P_A^a = \alpha^{k_A a}$, pouze z dostupných informací (α, I_A, u, P_A, p) spočítáním:

$$P_A \equiv \alpha^{h(I_A)} \cdot u^{-P_A} \bmod p.$$

5.2.2 Giraultův mechanismus se samocertifikovanými klíči

Kromě implicitně certifikovaných veřejných klíčů existují také samocertifikované veřejné klíče. Zde samocertifikovaný znamená, že uživatel sám může svůj soukromý klíč certifikovat bez pomoci další strany, tj. je jediný, kdo daný soukromý klíč zná.

Mechanismus spočívá v tom, že důvěryhodná strana T vytvoří implicitně certifikovaný, veřejně obnovitelný DH veřejný klíč pro stranu A, aniž by znala její soukromý klíč. Postup je následující:

Důvěryhodný server T zvolí tajná prvočísla p a q pro RSA modul $n = pq$, dále prvek $\alpha \in \mathbb{Z}_n^*$ maximálního řádu a nakonec příslušná čísla e a d (veřejný a soukromý RSA klíč) pro n. Každé straně A pak přidělí jednoznačně rozlišitelné jméno či identifikační řetězec I_A.

Strana A si pro sebe zvolí soukromý klíč a a spočítá veřejný klíč $\alpha^a \bmod n$ určený straně T. Navíc tím A prokáže T znalost odpovídajícího tajného klíče a.

Důvěryhodný server T spočítá veřejná data pro rekonstrukci jako $P_A = (\alpha^a - I_A)^d \bmod n$.

Odtud $(P_A^e + I_A) \bmod n = \alpha^a \bmod n$ a z veřejně dostupných informací může jakákoliv strana spočítat veřejný klíč α^a strany A.

5.3 DH protokoly s implicitně certifikovanými klíči

U těchto protokolů pro dohodu na klíči je autenticita veřejných klíčů zaručena právě rekonstrukcí z veřejně dostupných parametrů, tedy jsou implicitně certifikované. Takový postup lze aplikovat na další protokoly založené na DH, jako je DH s fixními exponenty, ElGamal či MTI/A0.

5.3.1 Güntherův protokol pro dohodu na klíči

Nejprve je potřeba definovat globální parametry: Za použití Güntherova mechanismu vytvoří důvěryhodný server T ElGamalův podpis (P_A, a) na identifikačním řetězci I_A a (P_B, b) na identifikačním řetězci I_B a tajně předá tyto podpisy stranám A a B spolu s parametry (p,α,u), kde p je prvočíslo, $\alpha \in \mathbb{Z}_p^*$ je generátor a u je veřejný klíč strany T.

Pokud jsou dlouhodobé parametry P_A a I_A stranám známy *a priori*, pak se tento protokol redukuje na dvoucestný.

Protokol:

---

1. $A \rightarrow B : I_A, P_A$

B: Vygeneruje náhodné číslo y, $1 \leq y \leq p-2$ a pošle následující zprávu.

2. $A \leftarrow B : I_B, P_B, (P_A)^y \bmod p$

A: Vygeneruje náhodné číslo x, $1 \leq x \leq p-2$ a pošle následující zprávu.

3. $A \rightarrow B : (P_B)^x \bmod p$

Výpočet klíče: A a B zkonstruují své veřejné klíče jako v Güntherově mechanismu, tedy (pořadě) $(P_B)^b$ a $(P_A)^a$ mod p, a strana A spočítá $K = (P_A^y)^a \cdot (P_B^b)^x$, strana B spočítá $K = (P_A^a)^y \cdot (P_B^x)^b$.

Společný klíč je tedy $K = \alpha^{k_A y a + k_B b x}$.

DH dohoda na klíči s fixním klíčem využívá implicitně certifikované klíče takto: A a B spočítají časově neměnný klíč $K = (\alpha^b)^a = (\alpha^a)^b$ mod n, získaný ze spočítaných hodnot $(P_B)^e + I_B$ mod $n = \alpha^b$, resp. $(P_A)^e + I_A$ mod $n = \alpha^a$ stran A a B v tomto pořadí.

Obdobně je tomu i u ElGamalova jedoprůchodového protokolu pro dohodu na klíči či u dvoucestného protokolu MTI/A0 (více v [1], str. 518).

Jiný způsob modifikace DH je použití Güntherových veřejných klíčů založených na ID (*identity-based*), u nějž jsou počáteční nastavení a značení stejné jako v protokolu výše. Zde strany A a B spočítají veřejné klíče α^{tb} a α^{ta} (v tomto pořadí) namísto hodnot α^b a α^a.

Jako poslední zmíním verzi těchto protokolů, zejména Güntherova založeného na ID, se samocertifikovanými klíči:

A vybere tajné náhodné číslo v, $1 \leq v \leq p-1$, $GCD(v, p-1) = 1$, spočítá $w = \alpha^v$ mod p a hodnotu w předá T. T vybere k_A a spočítá $P_A = w^{k_A}$ mod p (namísto α^{k_A} jako prve). Poté vyřeší rovnici (5.1) pro a a předá straně A dvojici $(r, s) = (P_A, a)$. Strana A pak spočítá $a' = a \cdot v^{-1}$ mod $(p-1)$. (P_A, a') je nyní ElGamalův podpis strany T identity I_A, což lze jednoduše ověřit: $u^{P_A} \cdot P_A^{a'} \equiv \alpha^{h(I_A)}$, a přitom T nezná a'.

PŘIJATO K OBHAJOBĚ 10.8.07

PŘEDSEDA KOMISE PRO BSZZ
STUDIJNÍ PROGRAM MATEMATIKA

Literatura

[1] Menezes A. J., Oorshot P. C. van, Vanstone S. A.: *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.

[2] Dostálek L., Vohnoutová M., *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*, Computer Press a.s., Brno, 2006.

[3] Boyd C., Mathuria A., *Protocols for Authentication and Key Establishment*, Springer, Berlin, 2003.

[4] Kaiser T., *Samoopravné kódy*, http://home.zcu.cz/~kaisert/kody/kody.pdf.

KVZ FAKULTY
oddělení