Karel Burda

Kryptografie okolo nás

Edice CZ.NIC

KRYPTOGRAFIE OKOLO NÁS
Karel Burda

Vydavatel:
CZ.NIC, z. s. p. o.
Milešovská 5, 130 00 Praha 3
Edice CZ.NIC
www.nic.cz

1. vydání, Praha 2019
Kniha vyšla jako 24. publikace v Edici CZ.NIC.
ISBN 978-80-88168-52-2

CZ.NIC | SPRÁVCE DOMÉNY CZ

— Karel Burda

Kryptografie okolo nás

— Edice CZ.NIC

Předmluva vydavatele

Předmluva vydavatele

Vážení čtenáři,

tím, jak se náš běžný život čím dál více přesouvá do online světa internetu, nabývá kryptografie na důležitosti. Do elektronického světa je nutné promítnout řešení problémů v reálném světě již alespoň částečně vyřešených, jako je zajištění soukromí při komunikaci, prokazování totožnosti komunikujících stran nebo autenticity vyměňovaných zpráv. Toto všechno a mnohem více pomáhá vyřešit právě kryptografie.

Vzhledem k tomu, že internet vznikal převážně na akademické půdě, nevnímali jeho průkopníci výše uvedené problémy jako klíčové, a tak spousta protokolů stojících v samotných základech internetu získávala důležité bezpečností prvky se značným zpožděním. Ještě větší urychlení v tomto směru přineslo odhalení praktik tajných služeb v posledních letech. Přestože použití kryptografie jako nástroje není možné z principu morálně posoudit, nelze ignorovat, že bývá nezřídka zneužívána k páchání trestné činnosti. Nechtěně se tak stává bitevním polem, na kterém se utkávají internetoví architekti a vývojáři aplikací na jedné straně a bezpečnostní složky nebo celé vlády na straně druhé. Bez ohledu na to, jak tato bitva dopadne, je určitě užitečné získat přehled o tom, kde všude se můžeme ve světě internetu s kryptografií setkat. A takový přehled vám zaručeně poskytne právě tato kniha.

Autor Karel Burda, vysokoškolský pedagog na VUT v Brně, touto knihou volně navazuje na své předchozí publikace a popisuje v ní celou řadu internetových technologií a protokolů, které kryptografii využívají. Osobně se s mnoha z nich setkávám při své práci denně. Jako příklad bych jmenoval DNSSEC (rozšíření protokolu DNS o kryptografické zajištění autenticity DNS odpovědí), v jehož rozšíření a tedy počtu zabezpečených domén jsme v ČR světovými lídry, nebo podporu nejmodernějšího autentizačního protokolu OpenID Connect službou mojeID, kterou u nás využívá přes 600 000 lidí.

Ať už knihu přečtete od začátku do konce, nebo pouze nalistuje část obsahující protokol, který vás nejvíce zajímá, jsem přesvědčen, že získané informace vám umožní pohybovat se po internetu s mnohem větší důvěrou a jistotou.

Jaromír Talíř, CZ.NIC
Praha, 1. listopadu 2019

Obsah

Úvod

Úvod

První kryptografické techniky (tzv. šifry) vznikly prakticky vzápětí po vzniku písma, neboť jejich účelem bylo utajit obsah písemných zpráv před nepovolanými čtenáři. Historicky nejstarším známým důkazem o praktickém využití kryptografie je hliněná destička ze starověké Mezopotámie z období asi 1.500 let před naším letopočtem, na níž je uveden šifrovaný popis technologie výroby glazurované keramiky. Zpočátku se v šifrách používaly jednoduché záměny znaků ve zprávě (tzv. substituce), nebo se ve zprávě měnilo pořadí znaků (tzv. transpozice). Postupně se však ukázalo, že z bezpečnostních důvodů je zapotřebí šifrovací postupy kombinovat a komplikovat. Šifry se tak stávaly stále složitějšími a k jejich použití, analýze a návrhu se proto začala používat matematika. V minulém století se také zjistilo, že matematické metody lze použít nejen k utajování obsahu zpráv, ale rovněž k prokazování původu doručených zpráv. Z původně jednoduchých šifrovacích technik se tak nakonec vyprofilovala kryptografie jako věda, která se zabývá konstrukcí a aplikací matematických metod pro utajování obsahu a prokazování původu přenášených zpráv.

Kryptografie primárně vznikla k ochraně zpráv během jejich přenosu, a tak až donedávna byly její doménou přenosové systémy. Praxe však ukázala, že pomocí kryptograficky chráněných zpráv lze velmi efektivně zajistit vysokou úroveň bezpečnosti mnoha dalších systémů, jako například systémů řízení přístupu, systémů elektronických plateb apod. S aplikacemi kryptografie proto přicházíme do styku každodenně, avšak všeobecné povědomí o tom, jak fungují, je nízká. Je to dáno zejména tím, že uvedené aplikace jsou poměrně složité.

Vzhledem ke složitosti a rozsáhlosti kryptografických aplikací budeme v této knize abstrahovat od různorodosti účelově stejných funkcí. To znamená, že celý soubor kryptografických funkcí, které jsou z hlediska svého účelu stejné, budeme reprezentovat jedinou obecnou funkcí. Nebudeme tak například rozlišovat, zda se šifrování provádí proudovou či blokovou šifrou, jakými algoritmy a případně v jakých režimech. Tímto přístupem si zjednodušíme situaci natolik, že nám pak k popisu naprosté většiny kryptografických aplikací postačí sedm elementárních kryptografických funkcí a jeden generátor. Konkrétně se jedná o:

- šifrovací (ENC) a k ní komplementární dešifrovací (DEC) funkci,
- pečetící (PCT) a k ní komplementární verifikační (VER) funkci,
- hešovací funkci (HSF),
- Diffie-Hellmanovu funkci (DHF),
- odvozovací funkci (ODF),
- generátor nepředvídatelných čísel (GEN).

Uvedený přístup umožní čtenáři rychle a efektivně porozumět principu fungování celé řady velmi různorodých kryptografických aplikací. Zaplatí za to určitými zjednodušeními, ale úplnější a detailnější informace se případný zájemce může dozvědět z odkazované literatury.

Struktura knihy je následující. Po tomto úvodu následuje kapitola Základy kryptografie, kde se čtenář podrobněji seznámí s výše uvedenými elementárními kryptografickými funkcemi. Zbytek knihy je věnován popisu kryptografických aplikací, s nimiž se potkáváme v běžném životě. Jsou rozděleny celkem do tří kapitol. V kapitole Přenosové systémy jsou vysvětleny kryptograficky zabezpečené protokoly, jejichž primárním účelem je zajistit bezpečný přenos zpráv. Příkladem jsou protokol TLS, který používáme kupříkladu v internetovém bankovnictví, nebo protokoly komplexu WPA, kterými se zabezpečují rádiové přenosy ve Wi-Fi sítích. V kapitole Přístupové systémy jsou popisovány protokoly k prokázání identity osob (např. Kerberos) a k řízení přístupu osob buď k síťovým službám (např. RADIUS), nebo do fyzických prostor (elektronická kontrola vstupu). Poslední kapitola s názvem Platební systémy je věnována platebním protokolům, jejichž účelem je elektronický převod peněz mezi účty uživatelů (např. protokoly pro platební karty či kryptoměna Bitcoin).

1 Základy kryptografie

1 Základy kryptografie

Jak již bylo uvedeno kryptografie je věda, která zkoumá matematické metody utajování obsahu i prokazování původu přenášených zpráv. Zprávou přitom budeme rozumět číselnou posloupnost, v níž je veřejně známým kódem zakódována informace. V praxi se zpravidla jedná o texty, obrázky a případně příkazy v podobě posloupností dvojkových číslic (tzv. bitů).

Autor zprávy (tzv. původce) svoji zprávu předává vhodným přenosovým systémem (typicky přes počítačovou síť) zamýšlenému příjemci (tzv. adresátovi). Z kryptografického hlediska mají běžně používané přenosové systémy charakter tzv. veřejného přenosového kanálu, což znamená, že ke kanálu mají kromě původce a adresáta přístup i jiné (tzv. neoprávněné) osoby. Některé z těchto osob usilují o čtení, resp. pozměňování přenášených zpráv, a tak je nazveme útočníky. Kryptografické techniky umožňují původci a adresátovi zajistit ochranu přenášených zpráv před uvedenými hrozbami. V případě utajování obsahu zpráv nejsou útočníci schopni zjistit, jaké zprávy jsou v přenosovém kanálu předávány (tzv. důvěrnost zpráv). A v případě prokazování původu zpráv si je adresát schopen ověřit, zda přijatá zpráva skutečně pochází od udávaného původce, tj. zda tato zpráva nebyla během přenosu případným útočníkem nějakým způsobem pozměněna, či dokonce nebyla celá podvržena (tzv. autentičnost zpráv).

Zabezpečená komunikace mezi původcem a adresátem je definována kryptografickým protokolem. Základ kryptografického protokolu tvoří datové jednotky, což jsou bloky bitů, které si mezi sebou původce s adresátem vyměňují. Každý typ datové jednotky má svoji určenou strukturu a svůj význam. Kromě různých typů datových jednotek je součástí protokolu i sada pravidel, jimiž se výměna datových jednotek mezi původcem a adresátem (v protokolu se nazývají strany) řídí. Kromě dvoustranných protokolů existují i vícestranné protokoly, což jsou protokoly, kde se výměna datových jednotek uskutečňuje mezi více subjekty. Reprezentantem vícestranného protokolu jsou například platební protokoly, na nichž se kromě plátce a příjemce zúčastňuje také banka plátce a banka příjemce.

Než přejdeme k základním kryptografickým funkcím, tak si zde ještě vysvětlíme operaci, která se nazývá zřetězení. Vstupem operace zřetězení jsou dva bitové bloky x a y, přičemž výstupem je blok z, který vznikne připojením bloku y na konec bloku x. Formálně budeme uvedenou operaci značit

$$z = x \| y.$$

Pokud máme například dva bloky $x = 00$ a $y = 101$, tak $z = x \| y = 00 \| 101 = 00101$.

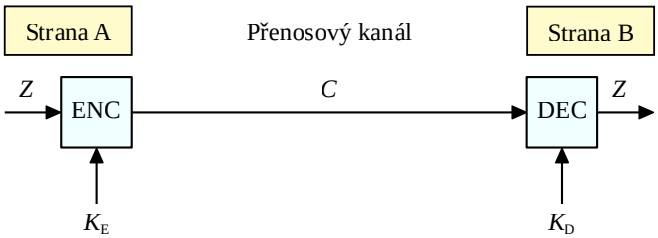
Na závěr této podkapitoly si ještě domluvíme typografické konvence, které v této knize použijeme. Jednotlivá písmena v základním řezu budou reprezentovat strany (např. strana A) a řetězce písmen v základním řezu budou vyjadřovat funkce (např. šifrování ENC). Písmena či jejich řetězce v kurzívě pak budou označovat proměnné (např. zpráva *Z* nebo blok ukončení zprávy *UZ*).

1.1 Utajovací kryptosystémy

Účelem utajovacích kryptosystémů je zajistit důvěrnost zpráv, což v praxi znamená utajení obsahu těchto zpráv před neoprávněnými osobami. Již jsme si uvedli, že zpráva (např. text či obraz) je číselná posloupnost, ve které jsou veřejně známým kódem zakódovány informace. K utajení obsahu zpráv (tedy k utajení zakódovaných informací) se zprávy Z převádějí na tzv. kryptogramy. Kryptogram C je číselná posloupnost, která je tzv. pseudonáhodná. To znamená, že kryptogram se jeví jako náhodná posloupnost čísel, avšak ve skutečnosti náhodnou posloupností není. Kryptogram je oproti zprávě veřejný, tj. je přístupný i neoprávněným osobám. Jeho pseudonáhodný charakter však způsobuje, že případní útočníci nejsou schopni jej převést zpět do podoby původní zprávy. Pouze oprávněná osoba, která zná tajný číselný parametr, je schopna provést konverzi kryptogramu na původní zprávu a z ní pak zakódované informace zjistit.

Strukturu utajovacího kryptosystému (nebo-li šifry) ilustruje obr. 1.1. Strana A (odesílatel) přivede svoji zprávu Z na vstup tzv. šifrovací funkce, kterou v dalším budeme označovat zkratkou ENC (z anglického „encrypting“). Tato funkce přiřazuje každé vstupní číselné posloupnosti Z právě jednu pseudonáhodnou posloupnost C. Z bezpečnostních důvodů musí v každém kryptosystému existovat velké množství použitelných šifrovacích funkcí (tzv. rodina funkcí) a konkrétní použitou funkci (tj. konkrétní přiřazení) pak definuje parametr K_E, který se nazývá šifrovací klíč. Přiřazení kryptogramu C vstupní zprávě Z pro danou hodnotu klíče K_E se nazývá šifrování a formálně je budeme vyjadřovat vztahem

$$C = \mathrm{ENC}(Z, K_E).$$



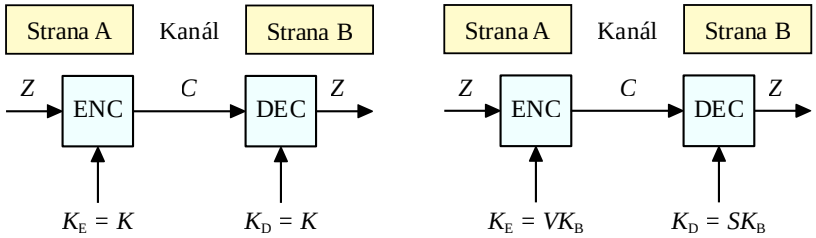
Obrázek 1.1: Struktura utajovacího kryptosystému

Odesílatel odešle kryptogram C veřejným kanálem k adresátovi B. Během tohoto přenosu se ke kryptogramu mohou dostat i neoprávněné osoby, avšak pouze oprávněný adresát B zná hodnotu tajného parametru K_D, který se nazývá dešifrovací klíč. Pomocí tohoto parametru nastaví tzv. dešifrovací funkci DEC („decrypting“) tak, aby byla inverzní k šifrovací funkci ENC. Na výstupu dešifrovací funkce se proto objeví původní zpráva Z. Formálně budeme proces dešifrování zapisovat vztahem

$$Z = \mathrm{DEC}(C, K_D).$$

Funkce ENC a DEC jsou obvykle veřejně známy (často jsou dokonce standardizovány). Požaduje se po nich, aby bez znalosti dešifrovacího klíče K_D nebylo prakticky možné ke kryptogramům C zjistit jim odpovídající zprávy Z. Další požadavek vyplývá ze skutečnosti, že dešifrovací klíč často slouží k dešifrování více kryptogramů. Proto se po funkcích ENC a DEC ještě požaduje, aby ze znalosti různých dvojic zpráva a odpovídající kryptogram nebylo prakticky možné odvodit hodnotu používaného dešifrovacího klíče K_D.

Utajovací kryptosystémy se dělí na symetrické a asymetrické (viz obr. 1.2).



Obrázek 1.2: Symetrický (vlevo) a asymetrický (vpravo) utajovací kryptosystém

U symetrických kryptosystémů platí, že zjistit hodnotu dešifrovacího klíče ze znalosti hodnoty klíče šifrovacího je prakticky možné. Z tohoto důvodu se musí utajovat hodnoty obou klíčů, přičemž obvykle platí, že tyto klíče jsou stejné, tj. $K_D = K_E = K$, kde K se nazývá tajný klíč. Symetrické kryptosystémy jsou rychlé, a tak se využívají k šifrování velkých objemů dat. Nevýhodou symetrických kryptosystémů je skutečnost, že bezpečné doručení klíče K komunikující protistraně je vzhledem k tajnému charakteru klíče komplikované.

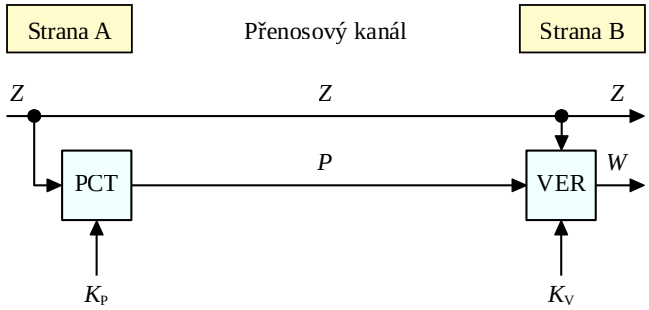
U asymetrických kryptosystémů naopak platí, že určení hodnoty dešifrovacího klíče ze znalosti hodnoty klíče šifrovacího je prakticky nemožné. Z tohoto důvodu je pak nutné utajovat pouze hodnotu dešifrovacího klíče. Adresát B si nejprve stanoveným postupem vytvoří dvojici šifrovací a dešifrovací klíč. Tato dvojice se odvozuje z velkých náhodných čísel, a tak pravděpodobnost, že dva uživatelé vytvoří stejnou dvojici klíčů, je prakticky rovná nule. Dešifrovací klíč je tajný a je znám pouze jeho tvůrci B (tzv. soukromý klíč SK_B strany B). Hodnotu šifrovacího klíče tvůrce daného kryptosystému zveřejní (tzv. veřejný klíč VK_B strany B). Platí tak, že $K_D = SK_B$ a $K_E = VK_B$. Adresátovi B potom může zasílat kryptogramy kdokoliv, bez nutnosti si s ním šifrovací klíč předem sjednat. Nevýhodou je, že asymetrické kryptosystémy jsou pomalé, a tak se používají k šifrování dat o malých objemech. Typicky se jedná o klíče pro symetrické kryptosystémy a o hesla.

1.2 Autentizační kryptosystémy

Účelem autentizačních kryptosystémů je garantovat adresátům původ doručených zpráv (tzv. autentičnost zpráv) [1]. Zpravidla se jedná o autorství zprávy, případně lze garantovat i další atributy původu zprávy, jako je čas nebo místo jejího vzniku. Princip autentizačních kryptosystémů je takový, že odesílatel spolu se zprávou Z odesílá i krátkou číselnou posloupnost P, která umožní adresátovi původ zprávy ověřit.

Strukturu autentizačního kryptosystému ilustruje obr. 1.3. Odesílatel A přivede svoji zprávu Z na vstup tzv. pečetící funkce, kterou budeme označovat zkratkou PCT. Zmíněná funkce přiřazuje každé vstupní číselné posloupnosti Z jednu z mnoha možných výstupních posloupností P. Posloupnost P nazveme pečeť. Konkrétní přiřazení pečetí jednotlivým zprávám se nastavuje pomocí tajného parametru K_P, jenž nazveme pečetící klíč. Proces přiřazení pečeti P vstupní zprávě Z budeme nazývat pečetění a formálně jej budeme vyjadřovat vztahem

$$P = \mathrm{PCT}(Z, K_P).$$



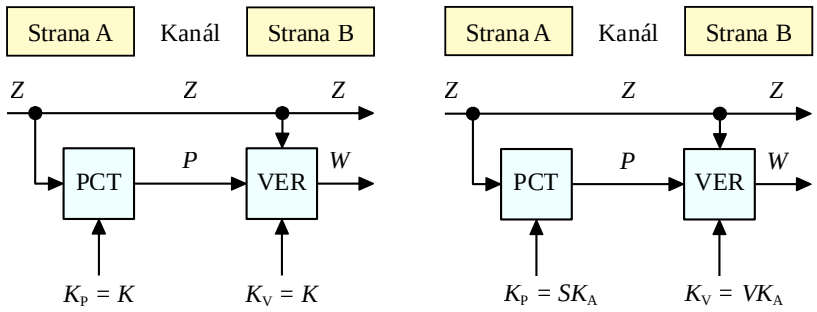
Obrázek 1.3: Struktura autentizačního kryptosystému

Dvojici (Z, P) nazveme zapečetěná zpráva. Odesílatel zapečetěnou zprávu odešle veřejným kanálem k adresátovi zprávy, přičemž během tohoto přenosu ji mohou neoprávněné osoby nahradit falešnou dvojicí (Z', P'). Adresát B zná správnou hodnotu parametru K_V (tzv. verifikační klíč), který definuje verifikační funkci VER. Přijatá zpráva Z spolu s její pečetí P jsou přivedeny na oba vstupy této funkce, v důsledku čehož výstupní hodnota W, kterou nazveme autentizační indikátor, má hodnotu buď 0, nebo 1. Verifikační funkce je konstruována tak, že pokud je pro danou zprávu pečeť správná, tak autentizační indikátor $W = 1$. V takovémto případě adresát zprávu Z akceptuje jako autentickou. V opačném případě, tj. když $W = 0$, adresát přijatou zprávu vyhodnotí jako podvodnou. Verifikaci budeme formálně zapisovat vztahem

$$W = \mathrm{VER}(Z, P, K_V).$$

Funkce PCT a VER jsou obvykle veřejně známy. Požaduje se po nich, aby bez znalosti pečetícího klíče K_P nebylo prakticky možné ke zprávám Z určit správné pečeti P. Pečetící klíč je zpravidla používán vícenásobně, tj. jeden pečetící klíč slouží k pečetění mnoha různých zpráv. Proto se také požaduje, aby ze znalosti různých dvojic zpráva a odpovídající pečeť nebylo prakticky možné odvodit hodnotu používaného pečetícího klíče K_P.

Stejně jako utajovací, tak i autentizační kryptosystémy se dělí na symetrické a asymetrické (viz obr. 1.4). U symetrických kryptosystémů platí, že zjistit hodnotu pečetícího klíče ze znalosti hodnoty klíče verifikačního je prakticky možné. Proto se musí utajovat hodnoty obou klíčů, přičemž obvykle platí, že tyto klíče jsou stejné, tj. $K_P = K_V = K$, kde K se nazývá tajný klíč. Pečeti symetrických autentizačních kryptosystémů se v anglicky psané literatuře označují mnoha různými zkratkami (MAC = „Message Authentication Code“, MIC = „Message Integrity Check“ nebo ICV = „Integrity Check Value“). Výhodou symetrických kryptosystémů je, že jsou rychlé, a proto se využívají při pečetění velkých množství zpráv. Jejich nevýhodou je opět fakt, že bezpečné doručení klíče K komunikující protistraně je vzhledem k tajnému charakteru klíče komplikované.



Obrázek 1.4: Symetrický (vlevo) a asymetrický (vpravo) autentizační kryptosystém

U asymetrických kryptosystémů naopak platí, že určení hodnoty pečetícího klíče ze znalosti hodnoty klíče verifikačního je prakticky nemožné. Z tohoto důvodu je tak nutné utajovat pouze hodnotu pečetícího klíče. Původce A si stanoveným postupem vytvoří dvojici pečetící a verifikační klíč. Uvedená dvojice se odvozuje z velkých náhodných čísel, a tak pravděpodobnost, že dva uživatelé vytvoří stejnou dvojici klíčů, je prakticky rovná nule. Pečetící klíč je tajný a je znám pouze jeho tvůrci (tzv. soukromý klíč SK_A strany A). Hodnotu verifikačního klíče tvůrce daného kryptosystému zveřejní (tzv. veřejný klíč VK_A strany A). Platí tedy, že $K_P = SK_A$ a $K_V = VK_A$. Autentičnost zpráv od strany A potom může ověřovat kdokoliv, bez nutnosti si s ním předem sjednat verifikační klíč.

Další velmi významnou předností autentizačních asymetrických kryptosystémů oproti symetrickým je skutečnost, že soukromý klíč SK_A zná pouze původce zprávy. A protože SK_A nelze ze

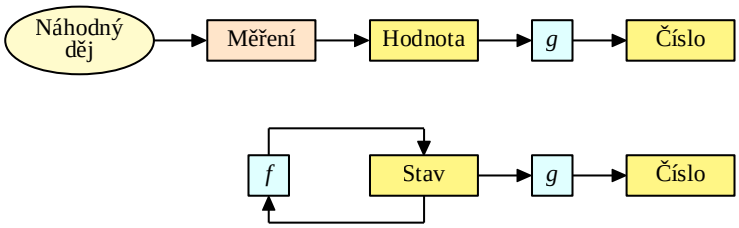
znalosti veřejného VK_A určit, tak správné pečeti dokáže vytvářet pouze strana A. Tato vlastnost způsobuje, že pečeti asymetrického autentizačního kryptosystému vykazují vlastnosti klasického ručního podpisu, kdy podpis dokáže vytvořit pouze jediná osoba a zároveň pravost tohoto podpisu může ověřit kdokoliv. Z tohoto důvodu se pečeti P u asymetrických kryptosystémů nazývají digitálními podpisy („Digital Signature“), soukromý klíč SK_A se nazývá podpisový klíč a funkce PCT se nazývá podpisová funkce. Nevýhodou podpisových kryptosystémů oproti autentizačním symetrickým kryptosystémům je, že jsou pomalé.

1.3 Generátory nepředvídatelných čísel

Generátory nepředvídatelných čísel jsou zařízení určená ke generování posloupností čísel, které se neoprávněným osobám jeví jako náhodné posloupnosti. To konkrétně znamená, že číselné hodnoty v generované posloupnosti mají stejnou pravděpodobnost výskytu a jednotlivá čísla posloupnosti se jeví, že jsou navzájem nezávislá. V kryptografii je tento typ generátorů používán zejména ke generování klíčů a ke generování unikátů (tj. čísel, jimiž se individualizuje vykonání a tedy i výstup kryptografické funkce). Jsou rovněž i základní komponentou tzv. proudových šifer. Vygenerování nepředvídatelného čísla N budeme formálně vyjadřovat

$$N = \mathrm{GEN}.$$

Generátory nepředvídatelných čísel klasifikujeme na náhodné a pseudonáhodné (viz obr. 1.5). Náhodné generátory (obr. nahoře) generují svá čísla na základě nějakého náhodného fyzikálního děje (např. tepelný šum). Měřením se zjišťuje aktuální hodnota náhodné veličiny a výsledky měření se vhodnou konverzní funkcí g převádějí na čísla pro výstup generátoru. Výstup těchto generátorů je tak skutečně náhodný („true random number generator“). Oproti tomu pseudonáhodné generátory („pseudorandom number generator“) generují svá čísla pomocí vhodného stavového automatu (obr. dole). Stavový automat je hardwarová (např. posuvný registr), nebo datová struktura (např. číslo), která může nabývat více stavů (např. obsah registru, nebo velikost čísla). Stavový automat se nejprve nastaví podle tajné náhodné hodnoty (tzv. semeno) do výchozího stavu. Pomocí přechodové funkce f se aktuálnímu stavu pokaždé přiřadí nový následující stav,



Obrázek 1.5: Generátor náhodných (nahoře) a pseudonáhodných (dole) čísel

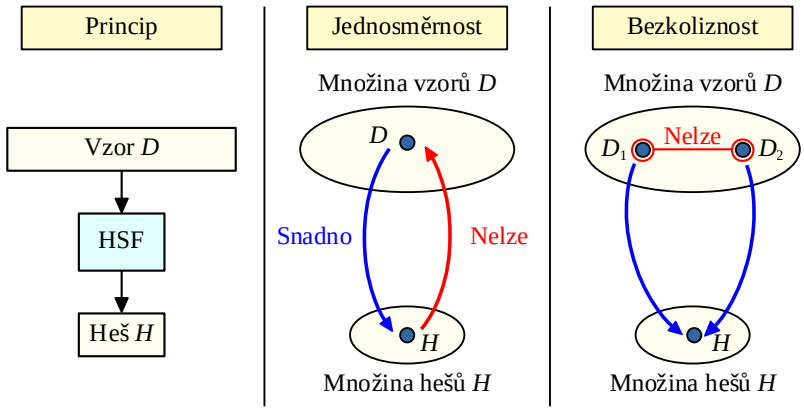
a tak automat postupně prochází všemi určenými stavy. Speciální konverzní funkce g přitom každému aktuálnímu stavu automatu přiřazuje číslo, které je výstupem generátoru. Chování pseudonáhodného generátoru je tedy zcela deterministické, tj. oprávněné osoby, které znají funkce f, g a semeno, dokáží vygenerovat tutéž posloupnost čísel. Požaduje se však, aby neoprávněná osoba došla statistickým testováním vygenerované posloupnosti k závěru, že daná posloupnost je náhodná.

1.4 Hešovací funkce

Hešovací funkce HSF je kryptografická funkce (viz obr. 1.6 vlevo), která číselnému argumentu D (nebo-li vzoru) o prakticky libovolné délce (jednotky bitů až triliony triliónů bitů) přiřazuje tzv. heš H, což je číselná hodnota o pevně stanovené délce (typicky o délce 256 až 512 bitů). Formálně budeme tuto funkci zapisovat

$$H = \mathrm{HSF}(D).$$

Od hešovací funkce se vyžadují dvě specifické vlastnosti, které se nazývají jednosměrnost a bezkoliznost. Jednosměrnost (obr. uprostřed) znamená, že určení hodnoty heše H je pro zadaný vzor D výpočetně snadné, avšak určení hodnoty vzoru D ze znalosti jeho heše H je prakticky nemožné. Bezkolizností (obr. vpravo) se rozumí, že je prakticky nemožné nalézt nějakou dvojici různých vzorů D_1 a D_2 takovou, aby jejich heše byly stejné. V této souvislosti je zapotřebí si uvědomit, že počet číselných posloupností libovolné délky (tj. počet vzorů) je vždy větší než počet posloupností jediné možné délky (tj. hešů). Z toho pak plyne, že mnoho vzorů musí mít stejný heš (tzv. kolize). Požaduje se však, aby nalezení kolize bylo prakticky nemožné.



Obrázek 1.6: Hešovací funkce a její vlastnosti

Jak v dalším uvidíme, tak jednosměrnost a bezkoliznost předurčuje hešovací funkce pro širokou škálu aplikací. Kromě vlastního samostatného nasazení jsou i důležitou komponentou zejména autentizačních kryptosystémů.

1.5 Diffie-Hellmanova funkce

Základním stavebním kamenem hojně využívaného Diffie-Hellmanova (zkráceně DH) protokolu je funkce, kterou nazveme Diffie-Hellmanova funkce (DHF). Uvedenou funkci (přesněji zobrazení) lze definovat pomocí různých typů konečných grup, avšak my se ve výkladu omezíme na číselné grupy s operací násobení. Tato operace nám dovoluje definovat mocnění, kdy například $V \cdot V \cdot V = V^3$, přičemž výchozí V i výsledné V^3 jsou čísla dané grupy. Abychom nemuseli vysvětlovat modulární aritmetiku, tak použijeme notaci bez operace modulo, čímž bude vysvětlení vlastností funkce DHF principiálně srozumitelné i čtenářům, kteří tuto operaci neznají. Podle avizované notace je pro argument V a tajný parametr a výstupní hodnotou Diffie-Hellmanovy funkce hodnota

$$A = \mathrm{DHF}(V, a) = V^a.$$

Uvedenou funkci lze interpretovat (viz [2]) jako specifickou symetrickou šifru, kde vstupní zprávou je číslo V, tajným klíčem je hodnota a a výstupem je kryptogram A. Diffie-Hellmanova funkce, jako každá šifra, se vyznačuje tou vlastností, že neoprávněná osoba není schopna ze znalosti vstupní hodnoty V a výstupního kryptogramu A zjistit hodnotu tajného klíče a. Oproti běžným šifrám má však Diffie-Hellmanova funkce jednu velmi zajímavou vlastnost. K její ilustraci si zvolme další tajný klíč b a jemu příslušející kryptogram

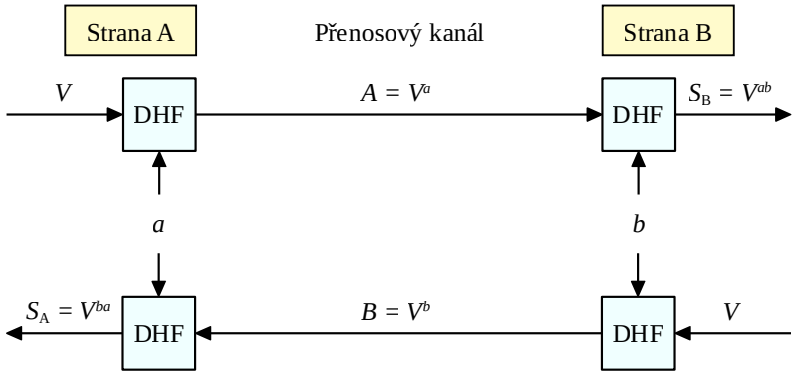
$$B = \mathrm{DHF}(V, b) = V^b.$$

Ze všeobecně známých vlastností mocnin pak pro Diffie-Hellmanovu funkci vyplývá, že

$$\mathrm{DHF}(B, a) = B^a = (V^b)^a = V^{b \cdot a} = (V^a)^b = A^b = \mathrm{DHF}(A, b).$$

Ze získané rovnosti vidíme, že pokud kryptogram B zašifrujeme tajným klíčem a (viz levá strana předchozí rovnice), tak vznikne zcela stejný výsledek, jako když kryptogram A zašifrujeme tajným klíčem b (pravá strana rovnice). Jinými slovy můžeme říci, že pokud vstup V zašifrujeme klíčem b, tak získáme kryptogram B, který když poté zašifrujeme klíčem a, tak získáme tentýž výsledek, jako když vstup V nejprve zašifrujeme klíčem a (výsledkem je kryptogram A) a poté klíčem b. Uvedenou vlastnost lze obecně formulovat tak, že výsledek série několika po sobě jdoucích šifrování různými klíči nezávisí na pořadí použitých klíčů, ale závisí jen na jejich součinu.

Nyní si můžeme popsat samotný DH protokol. Jeho účastníky jsou strany A a B (viz obr. 1.7).



Obrázek 1.7: Schéma Diffie-Hellmanova protokolu

Strana A si vygeneruje tajný klíč a a vypočítá A = DHF(V, a). Strana B si vygeneruje svůj klíč b a vypočítá B = DHF(V, b). Kryptogramy A a B si obě strany prostřednictvím veřejného kanálu vymění. Strana A z přijatého kryptogramu vypočítá hodnotu S_A = DHF(B, a) a strana B vypočítá S_B = DHF(A, b). Z rovnice u předchozího odstavce vyplývá, že DHF(B, a) = DHF(A, b), tj. platí, že $S_A = S_B = S$ = DHF(V, $a \cdot b$).

Diffie-Hellmanův protokol umožňuje stranám A a B sestrojit pomocí veřejného kanálu (tj. kanálu, který může být pod kontrolou neoprávněných osob) sdílenou tajnou hodnotu S (tzv. semeno – "seed"). Z této hodnoty pak lze pomocí funkce ODF (viz dále) odvodit tajné klíče pro symetrické kryptosystémy a těmito kryptosystémy další komunikaci mezi A a B ve veřejném kanálu chránit. Bezpečnost protokolu spočívá v tom, že neoprávněné osoby sice mohou v kanále zachytit oba kryptogramy A a B, avšak k sestrojení semena S_A, resp. S_B potřebují zjistit tajný klíč a, resp. b. Již jsme si však uvedli, že funkce DHF je konstruována tak, aby to ze známých hodnot V, A a B nebylo prakticky možné.

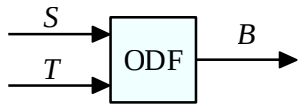
Poznamenáváme, že v Diffie-Hellmanově protokolu se klíče a a b, resp. kryptogramy A a B často nazývají soukromé, resp. veřejné klíče. To je však matoucí označení, neboť hodnoty A i B v tomto případě roli klíče neplní. Klíč má v kryptografických funkcích význam parametru, tj. údaje, který z rodiny všech možných přiřazení (např. rodiny všech možných šifrovacích přiřazení) určuje právě jedno konkrétní přiřazení (např. právě jedno konkrétní přiřazení výstupních kryptogramů vstupním zprávám). Pokud se však na obr. 1.7 podíváme například na funkci DHF vpravo nahoře, tak vidíme, že podle zmiňované terminologie má tato funkce celkem dva parametry (klíče A a b), ale žádný argument. To je logický nesmysl, a proto budeme používat jinou terminologii.

Navržená terminologie vychází z toho, že funkce DHF je v pravém slova smyslu zobrazením na grupě. Prvky této grupy mohou být buď čísla (tuto variantu jsme si popisovali), nebo body eliptické

křivky, a tak vstupní prvek V budeme obecně nazývat DH vzor a výstupní prvky A, resp. B budeme nazývat DH obraz strany A, resp. B. Semeno S je pak DH obrazem vzoru V vytvořeným společně oběma stranami a čísla a a b jsou tajné klíče jednotlivých stran. U některých popisů budeme používat konvenci, kdy strana X má tajný klíč K_X a pro obvykle veřejně známý DH vzor V má tato strana svůj DH obraz $Q_X = \mathrm{DHF}(V, K_X)$.

1.6 Odvozovací funkce

Odvozovací funkce ODF (obr. 1.8) primárně slouží k odvozování hodnot klíčů symetrických kryptosystémů na základě tajné hodnoty semena S a tzv. kontextu T. Kontextem se přitom rozumí neutajované (tj. veřejně známé) hodnoty, které mohou mít charakter konstant ale i variabilních hodnot. Kontexty slouží k individualizaci odvozovací funkce (např. podle účelu odvozeného klíče, či podle jejích uživatelů). Z důvodů vyšší bezpečnosti nebo z důvodu odvození různých druhů klíčů se odvozovací funkce mohou používat vícenásobně, takže výstupem odvozovací funkce může být i hodnota nového semene.



Obrázek 1.8: Odvozovací funkce

Obvyklé použití odvozovací funkce ODF je následující. Obě komunikující strany A a B si nejprve jednorázově sjednají společné semeno S. K tomu se nejčastěji využívá technika Diffie-Hellmanova protokolu, ale lze použít i techniku fyzického transportu semena v bezpečném paměťovém úložišti pomocí kurýra, nebo techniku přenosu semena šifrovacím kryptosystémem. V případě potřeby nového klíče (např. před navázáním komunikace, nebo po uplynutí určené doby používání klíče) obě strany vygenerují svá náhodná čísla N_A, resp. N_B a tato čísla si prostřednictvím veřejného kanálu navzájem vymění. Z obou čísel každá strana sestrojí společný kontext T (např. $T = N_A \parallel N_B$) a pomocí funkce ODF pak vygenerují stejný pseudonáhodný číselný blok B potřebné délky. Formálně pak

$$B = \mathrm{ODF}(S, T).$$

Získaný blok B obě strany podle stanovených pravidel rozdělí na klíče pro jednotlivé symetrické kryptosystémy. Obvykle jedná o šifrovací klíč KE_{AB} pro směr komunikace z A do B, klíč KE_{BA} pro šifrování ve směru z B do A, pečetící klíč KP_{AB} pro provoz z A do B a klíč KP_{BA} pro pečetění v opačném směru. Potom můžeme psát, že

$$\mathrm{ODF}(S, T) = B = KE_{AB} \parallel KE_{BA} \parallel KP_{AB} \parallel KP_{BA}.$$

Abychom si však popisy kryptosystémů co nejvíce zjednodušili, tak v dalším výkladu klíče podle směru komunikace rozlišovat nebudeme.

Výhodou funkce ODF je skutečnost, že při dlouhodobé platnosti semene *S* lze hodnoty klíčů pro symetrické kryptosystémy často a operativně měnit. Tímto způsobem se podstatně zvyšuje bezpečnost komunikace. Přirozeným požadavkem na odvozovací funkci je praktická nemožnost odvodit tajné semeno *S* na základě znalosti kontextu *T* a výstupního bloku *B*.

Z hlediska terminologie poznamenáváme, že především ve starších pramenech se semeno *S* nazývá klíč. Pro oba tyto pojmy je společné, že jsou tajné. Rozdílem však je, že semeno je argumentem funkce, kdežto klíč je parametrem funkce. Za poznámku rovněž stojí, že funkce ODF z hlediska svého účelu připomíná generátor pseudonáhodné posloupnosti. Semeno zde určuje počáteční stav generátoru a kontext hraje roli parametru, jímž se individualizuje fungování generátoru. Rozdílem je, že funkce ODF slouží ke generování pseudonáhodné posloupnosti pro blok klíčů i ze semena, které nemusí být ideálně náhodné (typicky DH obraz), avšak na druhou stranu ji nelze použít ke generování velmi dlouhých posloupností (např. pro proudovou šifru).

1.7 Kryptografické proměnné

S kryptografickými proměnnými jsme se již setkali a zde si je utřídíme. Kryptografické proměnné jsou číselné posloupnosti, jejichž variabilita zajišťuje vyšší úroveň bezpečnosti kryptosystémů (například cestou časté změny hodnot tajných klíčů) a zároveň také zajišťuje unikátnost kryptografických systémů různých uživatelů (například každý uživatel může mít svůj unikátní soukromý klíč).

Kryptografické proměnné lze podle požadavku na jejich utajení klasifikovat následovně.

- Utajované proměnné:
  - tajné klíče symetrických kryptosystémů,
  - soukromé klíče asymetrických kryptosystémů,
  - semena.
- Veřejné proměnné:
  - veřejné klíče asymetrických kryptosystémů,
  - unikáty.

Kryptografické funkce přiřazují každému argumentu (tj. každé hodnotě z množiny vstupních hodnot) právě jednu výstupní hodnotu. Konkrétní přiřazení je definováno pomocí klíče. Klíče tak jsou parametry, které z množiny všech možných přiřazení (z tzv. rodiny funkcí) definují jedno konkrétní přiřazení (funkci). Dalším typem proměnné je semeno, což je tajná hodnota, která je argumentem kryptografické funkce (např. ODF), případně počátečním stavem pseudonáhodného generátoru. Poslední kryptografickou proměnnou je unikát, což je veřejně známý argument,

kterým se individualizuje vykonání (a tedy i výstup) kryptografické funkce. Unikátem může být náhodné číslo, aktuální čas, nebo pořadové číslo.

1.8 Ustavení klíčů

Bezpečnost naprosté většiny moderních kryptosystémů je budována na pesimistickém předpokladu, že útočník ví o daném kryptosystému úplně vše, kromě tajného, resp. soukromého klíče. Ustavení klíčů (tj. bezpečné získání klíčů komunikujícími stranami) je proto kritickým problémem každého kryptosystému.

U symetrických kryptosystémů jsme si uvedli, že obě strany disponují tajným klíčem K. Tento klíč je možné ustavit více způsoby. První možností je transport klíče kurýrem, druhou možností je šifrovaný transport klíče veřejným kanálem a třetí možností je sestrojení klíče Diffie-Hellmanovým protokolem. Technika transportu kurýrem je založena na fyzické přepravě speciálního paměťového úložiště („key loader"). Klíč K se jednoduše do tohoto transportního úložiště zapíše a spolehlivý kurýr doručí úložiště komunikující straně X. Tato strana se vůči úložišti stanoveným způsobem autentizuje (např. heslem), a to následně dovolí přečtení klíče ze své paměti.

Druhou možností je šifrovaný transport klíče K pomocí veřejného kanálu, a to buď symetrickým, nebo asymetrickým kryptosystémem. V případě symetrického kryptosystému mají obě strany sjednán tzv. transportní klíč KT, který je vyhrazen pouze pro transport klíčů. Tento klíč má dlouhodobou platnost a je obvykle doručen kurýrem. Postup je takový, že například strana A vygeneruje hodnotu klíče K pomocí generátoru náhodných čísel (tj. $K = \mathrm{GEN}$), tento klíč zašifruje transportním klíčem do podoby kryptogramu $C = \mathrm{ENC}(K, KT)$ a kryptogram veřejným kanálem předá protistraně B. Ta kryptogram dešifruje a získá tak klíč $K = \mathrm{DEC}(C, KT)$. Ten pak obě strany používají k šifrovanému přenosu zpráv.

V případě asymetrického kryptosystému je postup obdobný. Například opět strana A vygeneruje klíč K pomocí generátoru náhodných čísel (tj. $K = \mathrm{GEN}$). Tento klíč však nyní zašifruje veřejným klíčem protistrany B (tj. klíčem VK_B) do podoby kryptogramu $C = \mathrm{ENC}(K, VK_B)$ a kryptogram C veřejným kanálem předá straně B. Ta kryptogram dešifruje svým soukromým klíčem SK_B, čímž získá klíč $K = \mathrm{DEC}(C, SK_B)$. Klíč K pak obě strany opět používají k šifrovanému přenosu zpráv.

Třetí možností je sestrojení klíče pomocí Diffie-Hellmanova protokolu. Pomocí tohoto protokolu (viz podkapitola 1.5) si obě strany sjednají tajné semeno S, navzájem předají své unikáty a z těchto hodnot následně pomocí odvozovací funkce ODF (viz podkapitola 1.6) odvodí klíč K.

V případě ustavení klíčů u asymetrických kryptosystémů je zapotřebí rozlišovat mezi soukromým a veřejným klíčem. U soukromého klíče žádný problém s jeho ustavením není. Pokud si například strana B vygeneruje dvojici soukromý klíč SK_B a veřejný klíč VK_B, tak svůj soukromý klíč nikomu nemusí poskytovat – klíč SK_B bezpečně uloží na vhodné paměťové úložiště a používá jej podle

potřeby. V případě veřejného klíče se ohledně jeho ustavení zdánlivě o žádný problém rovněž nejedná. Klíč VK_B je přece veřejný, a tak jej lze přenášet veřejným kanálem. Vzniká se tu však problém autentičnosti klíče. Pokud je například straně A doručena zpráva Z, v níž se uvádí, že strana B má veřejný klíč VK_B, tak sice může jít o pravdivou zprávu, ale stejně tak může jít o zprávu podvodnou. Podvod spočívá v tom, že si útočník U vygeneruje svoji dvojici soukromý klíč SK_U a veřejný klíč VK_U, přičemž straně A zašle klíč VK_U s tvrzením, že se jedná o veřejný klíč strany B. Strana A pak tímto klíčem bude šifrovat zprávy, resp. ověřovat podpisy a přitom si bude myslet, že bezpečně komunikuje se stranou B. Ve skutečnosti však bude komunikovat s útočníkem U. K eliminaci popsaného podvodu vznikly tzv. certifikační autority.

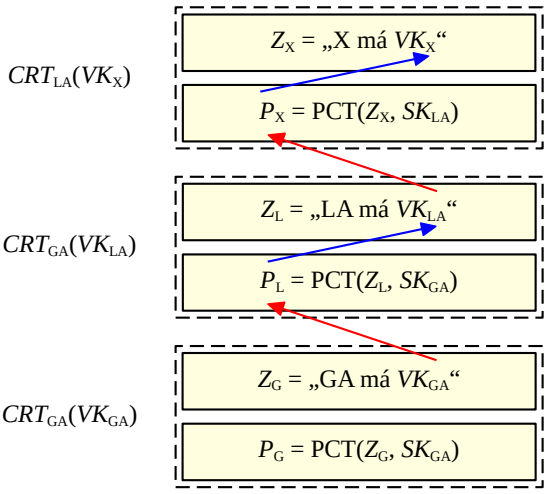
Certifikační autorita CA je důvěryhodná strana, která vydává tzv. certifikáty. Zájemce o certifikát nejprve certifikační autoritě prokáže svoji identitu X a vlastnictví veřejného klíče VK_X. Ta mu pak vydá certifikát, což je prakticky autoritou podepsaná zpráva Z_X = „Strana X má klíč VK_X“. Certifikát strany X vydaný certifikační autoritou CA je tedy dvojice (Z_X, P_X), kde podpis $P_X = \mathrm{PCT}(Z_X, SK_{CA})$ a SK_{CA} je soukromý podepisovací klíč CA. Uvedený certifikát budeme zkráceně zapisovat jako $CRT_{CA}(VK_X)$.

Veřejný klíč VK_{CA}, který stranám slouží k ověřování certifikátů certifikační autority CA, je obvykle distribuován ve formě tzv. kořenového certifikátu $CRT_{CA}(VK_{CA})$, což je certifikát, který autorita podepsala sama sobě. Platí tady, že $CRT_{CA}(VK_{CA}) = (Z_{CA}, P_{CA})$, přičemž zpráva Z_{CA} = „CA má klíč VK_{CA}“ a podpis $P_{CA} = \mathrm{PCT}(Z_{CA}, SK_{CA})$. Ostatní strany tento kořenový certifikát získají bezpečným způsobem (např. osobní návštěvou certifikační autority). Pokud pak strana Y disponuje takovýmto kořenovým certifikátem a strana X ji zašle svůj certifikát $CRT_{CA}(VK_X)$, tak si strana Y může klíčem VK_{CA} z kořenového certifikátu ověřit, že podle certifikační autority protistrana X skutečně existuje a že disponuje veřejným klíčem VK_X. Jednorázovým bezpečným získáním jediného klíče (klíče VK_{CA}) nyní může každá strana bezpečně ustavit veřejný klíč s například všemi desetitisíci stranami, jimž certifikační autorita certifikát vydala.

Aby jednotlivé strany nemusely ve svém paměťovém úložišti uchovávat velký počet kořenových certifikátů, tak byla do světa certifikačních autorit zavedena hierarchie. Tuto hierarchii si budeme ilustrovat na jednoduché dvoustupňové hierarchii, v níž budeme rozeznávat hierarchicky vyšší autority (nazveme je globální – GA) a hierarchicky nižší (tzv. lokální – LA) autority. V popsaném uspořádání pak platí, že lokální autority podepisují certifikáty pro komunikující strany a globální autority podepisují certifikáty pro lokální autority. Na obrázku 1.9 máme příklad dvoustupňové hierarchie, v němž globální autorita GA (např. DigiCert) vydala lokální autoritě LA (např. certifikační autoritě banky B) certifikát $CRT_{GA}(VK_{LA})$. Lokální autorita LA přitom vydává certifikáty zařízením své banky, přičemž konkrétně serveru X vydala certifikát $CRT_{LA}(VK_X)$. Jak dále uvidíme, tak potom každé straně Y, která chce bezpečně komunikovat se zařízeními banky B, nyní postačí mít pouze kořenový certifikát $CRT_{GA}(VK_{GA})$ globální certifikační autority a nepotřebuje bezpečně získat kořenový certifikát lokální certifikační autority.

Při zahájení komunikace server X protistraně Y zašle certifikáty $CRT_{GA}(VK_{LA})$ a $CRT_{LA}(VK_X)$. Jejich napojením na kořenový certifikát $CRT_{GA}(VK_{GA})$ vznikne pro stranu Y souvislý řetězec

certifikátů. Strana Y si nejprve z kořenového certifikátu ověří správnost klíče VK_{GA}. S jeho pomocí si pak ověří podpis P_L z $CRT_{GA}(VK_{LA})$ (červená šipka od VK_{GA} k P_L). A protože strana Y důvěřuje GA, tak potom věří i jí podepsané zprávě, že „LA má VK_{LA}“ (modrá šipka k VK_{LA}). Pomocí tohoto VK_{LA} ověří podpis P_X z $CRT_{LA}(VK_X)$ (červená šipka od VK_{LA} k P_X). A pokud se strana Y rozhodne důvěřovat i LA, tak bude věřit jí podepsané zprávě, že „X má VK_X“ (modrá šipka k VK_X). Tímto způsobem si libovolná strana Y může relativně bezpečně ověřit, že server X skutečně existuje a že disponuje klíčem VK_X. Řetězce certifikátů mohou v praxi sestávat i z většího počtu certifikátů (vícestupňová hierarchie). Výhodou techniky řetězení je, že s několika bezpečně získanými kořenovými certifikáty, lze ustavit veřejné klíče prakticky s jakýmkoliv dalším zařízením na této planetě.



Obrázek 1.9: Řetězec certifikátů

2 Přenosové systémy

2 Přenosové systémy

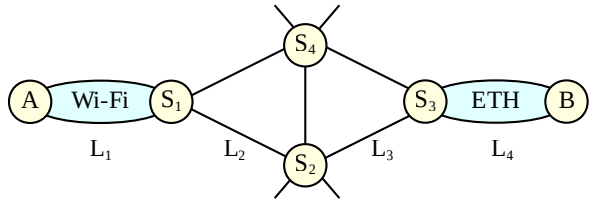
Přenosové systémy jsou systémy určené k přenosu zpráv mezi stranami. Z historie známe celou řadu přenosových systémů (telefonní, dálnopisné atd.), ale v současné době jsou dominantními přenosovými systémy počítačové sítě, které jsou založeny na síťovém protokolu IP („Internet Protocol“). Tyto systémy budeme v dalším označovat IP sítě. K pochopení jejich kryptografického zabezpečení si nejprve musíme vysvětlit fungování a architekturu tohoto typu sítí.

2.1 IP sítě

Již jsme si uvedli, že zpráva je číselnou posloupností konečné délky, v níž je zakódována informace. V počítačových sítích se jedná o bloky bitů, jejichž původcem, resp. adresátem je počítačový program (tzv. aplikace). To platí i v případě, že skutečným původcem, resp. adresátem je osoba. Aplikace totiž na jedné straně umožňuje osobě informaci do podoby bloku bitů zakódovat (např. textový editor) a aplikace na protistraně umožňuje zakódovanou informaci osobě prezentovat (např. prohlížeč textu).

Počítačové sítě jsou velmi rozsáhlé a složitě provázané systémy. Zprávy mezi aplikacemi, které běží na mnoha různých zařízeních, je zapotřebí přenášet současně a zároveň na velké vzdálenosti. Splnění uvedených požadavků se ukázalo jako složitý problém, který nakonec musel být rozčleněn na několik dílčích problémů. Zmíněnou dekompozici definuje tzv. síťový model, jenž člení síťovou architekturu počítačové sítě do několika hierarchicky uspořádaných vrstev. V takovéto architektuře platí, že mezi protokoly sousedících vrstev existují přesně definovaná rozhraní a že protokol hierarchicky vyšší vrstvy využívá služby, které poskytují protokoly v jemu podřízené vrstvě. Obě tyto zásady zajišťují, že v každé vrstvě lze vyvíjet nové a inovovat stávající protokoly bez toho, že by si tyto změny vynucovaly změny v ostatních vrstvách architektury. V IP sítích je definován čtyřvrstvý model, kde hierarchicky nejvyšší je aplikační vrstva, pod ní se nachází transportní vrstva, ještě níže leží síťová vrstva a zcela nejníže se nachází linková vrstva.

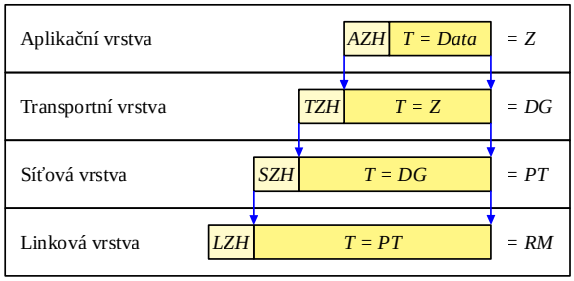
Protokoly aplikační vrstvy řeší komunikaci mezi aplikacemi, tj. definují významy a formáty zpráv a definují pravidla výměny těchto zpráv. K nejpoužívanějším protokolům aplikační vrstvy náleží zejména protokol pro webovou komunikaci HTTP („Hypertext Transfer Protocol“) a protokol pro elektronickou poštu SMTP („Simple Mail Transfer Protocol“). K fyzickému přenosu zpráv mezi aplikacemi slouží zbývající tři vrstvy. K pochopení jejich úlohy použijeme příklad na obr. 2.1.



Obrázek 2.1: Příklad IP sítě

Podle tohoto obrázku komunikuje aplikace, která běží na uživatelském počítači A (např. webový prohlížeč), s aplikací běžící na počítači B (např. webový server). Počítač A je připojen přes rádiovou síť Wi-Fi ke směrovači S_1. Směrovače jsou propojovacími prvky IP sítí a jejich funkci si vysvětlíme za chvíli. Směrovač S_1 je zároveň součástí tzv. internetové sítě, což je síť navzájem propojených směrovačů (na obrázku máme segment této sítě sestávající ze směrovačů S_1 až S_4). Směrovač S_3 je i přípojným bodem pro ethernetovou síť (ETH), do níž je připojen cílový počítač B. Jak dvoubodové spoje mezi směrovači (např. S_1 – S_2), tak i celé ethernetové či Wi-Fi sítě jsou v rámci architektury IP sítí považovány za tzv. linky. Linkou se ve zmíněné architektuře rozumí technický systém, který zajišťuje přenos datových jednotek protokolu IP (tzv. paketů). Na obrázku máme označeny linky L_1 až L_4, které tvoří posloupnost použitou pro přenos zpráv mezi aplikacemi na počítačích A a B.

Datové jednotky protokolů různých vrstev obecně sestávají z tzv. záhlaví *ZH* a z těla *T*. Do záhlaví se uvádějí služební údaje potřebné pro zpracování datové jednotky v dané vrstvě a tělo tvoří buď samotná data určená k přenosu, nebo datová jednotka obvykle z nadřízené vrstvy. Předpokládejme nyní (viz obr. 2.2), že aplikace v počítači A vytvořila zprávu *Z*, která je určena pro aplikaci v počítači B. Zpráva sestává z aplikačního záhlaví *AZH*, kam aplikace uvádí služební údaje pro cílovou aplikaci a z těla *T*, které obsahuje samotná data. Zpráva *Z* spolu s IP adresou počítače B je z aplikační vrstvy předána do transportní vrstvy. V tomto příkladu budeme předpokládat, že aplikace používá k transportu zpráv protokol UDP („User Datagram Protocol“), jehož datovou jednotkou je tzv. datagram *DG*. Tělo *T* datagramu tvoří zpráva *Z* a transportní záhlaví *TZH* obsahuje služební údaje potřebné pro přenos zprávy transportní vrstvou. Kromě jiného se v tomto záhlaví nachází adresa cílové aplikace. Datagram je poté spolu s IP adresou počítače B předán síťové vrstvě.

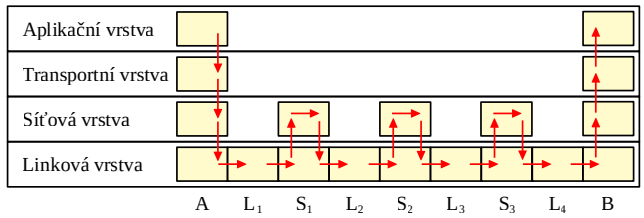


Obrázek 2.2: Datové jednotky protokolů v architektuře IP sítí

V síťové vrstvě je datagram *DG* uložen do těla paketu *PT*, což je datová jednotka protokolu IP. V síťovém záhlaví *SZH* paketu se uvedou služební údaje potřebné pro přenos síťovou vrstvou, což je zejména IP adresa cílového počítače B. Poté je paket *PT* předán linkové vrstvě. Ta paket uloží do těla rámce *RM*, což je datová jednotka linkového přenosového protokolu (v našem příkladu se jedná o Wi-Fi rámec). Do linkového záhlaví *LZH* rámce se zapíší služební údaje potřebné pro přenos linkovou vrstvou, což je v našem příkladě zejména linková adresa směrovače S_1.

Následně rámec *RM* opustí odesílající počítač A (viz obr. 2.3). Prostřednictvím síťové karty počítače A je vyslán do Wi-Fi sítě (z pohledu IP architektury do linky L_1), která podle cílové linkové adresy v *LZH* zajistí doručení rámce *RM* směrovači S_1. V tomto směrovači linková vrstva vyjme z doručeného rámce paket *PT* a předá jej do síťové vrstvy. Tam se ze síťového záhlaví *SZH* zjistí cílová IP adresa paketu (tj. počítač B) a podle řídících údajů síťové vrstvy se určí (tzv. směrování), že paket je nutno předat linkou L_2 do směrovače S_2. Paket je spolu s touto informací vrácen zpět do linkové vrstvy a vložen do nového rámce, v jehož záhlaví je nyní uvedena linková adresa S_2. Takto vzniklý rámec je linkou L_2 doručen směrovači S_2. Tam se postupuje obdobně jako v S_1, což nakonec vede k tomu, že rámec s paketem *PT* je linkou L_3 doručen směrovači S_3. I zde se provedou podobné procedury a výsledkem pak je, že linkový rámec je linkou L_4 doručen cílovému počítači B. Zde se z rámce vyjme paket *PT* a ten se předá síťové vrstvě. V síťové vrstvě se podle cílové IP adresy v záhlaví paketu zjistí, že paket již dorazil do cílového počítače. Z paketu se vyjme datagram a ten se předá transportní vrstvě. Z těla datagramu se vyjme zpráva *Z* a z transportního záhlaví *TZH* se zjistí adresa cílové aplikace. Té se zpráva *Z* předá.

Pokud si předchozí popis fungování IP sítě zrekapitulujeme, tak můžeme konstatovat, že účelem linek (tj. linkové vrstvy) je přenos paketů mezi zařízeními IP sítě, úkolem síťové vrstvy je přenos paketů posloupností linek od zdrojového k cílovému IP zařízení (takovýto štafetový transport paketů IP sítí nazveme IP spojením) a účelem transportní vrstvy je využití IP spojení pro přenos zpráv mezi aplikacemi.



Obrázek 2.3: Přenos dat IP sítí

2.2 Kryptografie v IP sítích

V moderních IP sítích jsou jednotlivé vrstvy architektury postupně doplňovány kryptografickými standardy s cílem zajistit jejich bezpečné fungování. Protože počet kryptograficky zabezpečených protokolů roste a jejich kategorizace je dosti nepřehledná, tak v této knize zavedeme kategorizaci založenou na následujících kritériích. Do kryptografických protokolů aplikační vrstvy zařadíme ty protokoly, které jsou součástí jedné jediné aplikace. Protokoly, které poskytují kryptograficky zabezpečený transport dat a využívá je více různých aplikací, tak ty zařadíme do transportní vrstvy. Síťové přenosové protokoly, respektive linkové přenosové protokoly, které využívají kryptografické techniky zabezpečení, pak zařadíme do síťové, respektive linkové vrstvy.

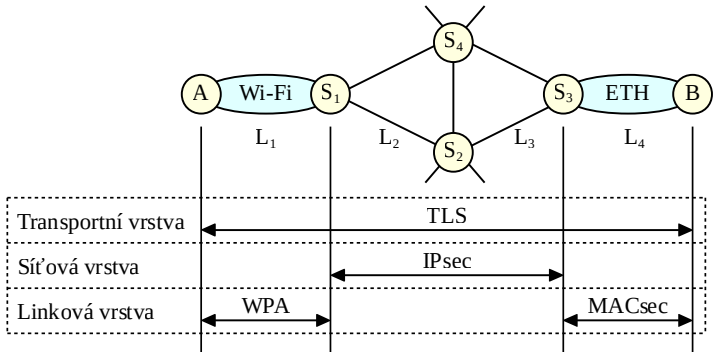
V linkové vrstvě se zejména jedná o protokoly komplexu WPA („Wi-Fi Protected Access"), které jsou určeny k zabezpečení spojů Wi-Fi sítí (v našem předešlém příkladu linka L_1) a protokol MACsec („Medium Access Control Security"), který je předurčen pro nasazení v ethernetových sítích (v našem příkladu linka L_4). Jiné typy spojů (např. spoje synchronní či plesiochronní digitální hierarchie) lze často zabezpečit pro ně určenými linkovými šifrátory.

V síťové vrstvě je základním přenosovým protokolem protokol IP („Internet Protocol"), přičemž k jeho zabezpečení se využívá kryptografická nadstavba IPsec („Internet Protocol Security"). Do protokolů síťové vrstvy zařadíme i protokol anonymizační sítě Tor. V transportní vrstvě se v praxi nejvíce využívají transportní protokoly TCP („Transmission Control Protocol") a již zmiňovaný protokol UDP („User Datagram Protocol"), avšak uvedené protokoly nejsou kryptograficky nijak zabezpečeny. K bezpečnému transportu dat od aplikace na jednom počítači k aplikaci na jiném počítači je určen protokol TLS („Transport Layer Security") a protokol SSH („Secure Shell"). Oba tyto protokoly jsou fakticky kryptografickou nadstavbou protokolu TCP. Ke kryptografickému zabezpečení protokolu UDP se v současné době experimentuje s protokolem DTLS („Datagram Transport Layer Security").

S kryptograficky zabezpečenými protokoly se lze setkat také v aplikační vrstvě (např. DNSsec, nebo elektronická pošta). Obvykle však aplikační protokoly využívají ke svému zabezpečení kryptografická řešení nižších vrstev, zejména pak transportní vrstvy. Využívá se přitom skutečnost, že zabezpečením (např. zašifrováním) datové jednotky v určité vrstvě jsou automaticky zabezpečeny i všechny do ní vložené datové jednotky vyšších vrstev (viz obr. 2.2). Z tohoto úhlu pohledu je potom optimální řešit kryptografické zabezpečení v linkové vrstvě. Nevýhodou takovéhoto řešení však je, že kryptografické zabezpečení je omezeno pouze na jednotlivou linku. Naproti tomu kryptograficky zabezpečené protokoly vyšších vrstev (tj. síťové, transportní, či aplikační) mohou zajistit bezpečný přenos dat přímo od zdrojového počítače až po cílový počítač.

Vzhledem k nezávislosti jednotlivých vrstev architektury IP sítí lze provozovat kryptografické protokoly v různých vrstvách souběžně bez toho, že by se navzájem nějak ovlivňovaly. V našem příkladu z obr. 2.1 pak můžeme kryptografické zabezpečení našich přenosů navrhnout tak, jak je uvedeno na obr. 2.4.

Na linkové vrstvě je v případě Wi-Fi sítě (linka L_1) použit protokol WPA a v ethernetové síti (linka L_4) je nasazen protokol MACsec. K ochraně komunikace vedené přes internetovou síť (v našem příkladu přes linky L_2 a L_3) je mezi směrovači S_1 a S_3 použit protokol IPsec, tj. kryptografický protokol v síťové vrstvě). A pro vybrané aplikace na vybraných počítačích (v našem příkladu pro aplikace na počítačích A a B) je jejich vzájemná komunikace navíc zabezpečena ještě v transportní vrstvě a to protokolem TLS. Tím jsme zajistili, že k útoku na komunikaci mezi počítači A a B musí útočník překonat ve kterékoliv lince IP sítě vždy dva zcela nezávislé kryptografické protokoly. Podstatně se tak zvyšuje bezpečnost dané komunikace.



Obrázek 2.4: Příklad kryptografického zabezpečení IP sítě

V následujících podkapitolách si vybrané kryptografické protokoly IP sítí vysvětlíme. Před tím si však ještě vysvětlíme integrované symetrické kryptosystémy.

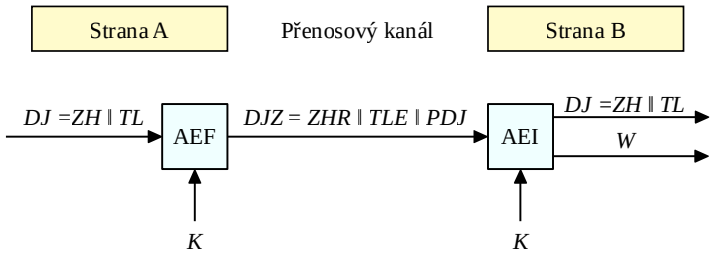
2.3 Integrované symetrické kryptosystémy

V nejnovějších verzích kryptografických protokolů různých vrstev (např. TLS 1.3 nebo ve WPA3) se prosazují symetrické kryptosystémy, které v sobě integrují utajovací a autentizační kryptosystém („Authenticated Encryption" – AE). Výhodou této integrace je vyšší bezpečnost přenášených dat, výpočetní efektivnost a jednodušší správa klíčů (k utajení i autentizaci se používá jeden společný klíč). Uvedenou kombinaci nazveme integrovaný symetrický kryptosystém.

Jak již víme, v počítačových sítích se data přenášejí v datových jednotkách (rámce, pakety apod.), které sestávají ze záhlaví a těla. V záhlaví datové jednotky se přenášejí služební data (typicky adresa příjemce) a v těle se přenášejí uživatelská data. Služební data utajovat sice nelze, protože síťová zařízení tato data potřebují k řízení pohybu datové jednotky sítí, ale na druhou stranu autentizace služebních dat zajistí eliminaci mnoha typů útoků. Proto integrované kryptosystémy často zajišťují nejen důvěrnost a autentičnost uživatelských dat, ale také autentičnost těch služebních dat, které se během přenosu nemění.

Pro popis integrovaného kryptosystému si zavedeme autentizační a pečetící funkci AEF a k ní inverzní funkci AEI (viz obr. 2.5). Kryptografické zabezpečení datové jednotky pak můžeme vyjádřit následovně. Má-li se přenést datová jednotka $DJ = ZH \parallel TL$, kde ZH je záhlaví a TL je tělo datové jednotky, tak integrovaný kryptosystém vytvoří zabezpečenou datovou jednotku $DJZ = ZHR \parallel TLE \parallel PDJ$, kde ZHR je rozšířené záhlaví (to obsahuje navíc služební kryptografické údaje), TLE je zašifrované tělo a PDJ je pečeť celé datové jednotky, tj. uživatelských a vybraných služebních dat. Odesílající strana zabezpečenou datovou jednotku DJZ formálně vypočítá tak, že $DJZ = \mathrm{AEF}(DJ, K)$,

kde K je klíč. Zabezpečená datová jednotka je zaslána protistraně, která provede inverzní operace a vypočítá dvojici $(DJ, W) = \mathrm{AEI}(DJZ, K)$. Pokud autentizační indikátor $W = 1$, tak je výstupní datová jednotka DJ považována za autentickou a je předána k dalšímu zpracování.



Obrázek 2.5: Schéma integrovaného symetrického kryptosystému

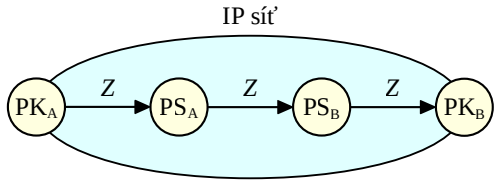
2.4 Kryptografie v aplikační vrstvě

Nejznámějšími reprezentanty kryptografie v aplikační vrstvě je zabezpečení elektronické pošty, protokol IKE a protokol DNSsec. Zmíněné protokoly si nyní vysvětlíme.

2.4.1 Zabezpečení elektronické pošty

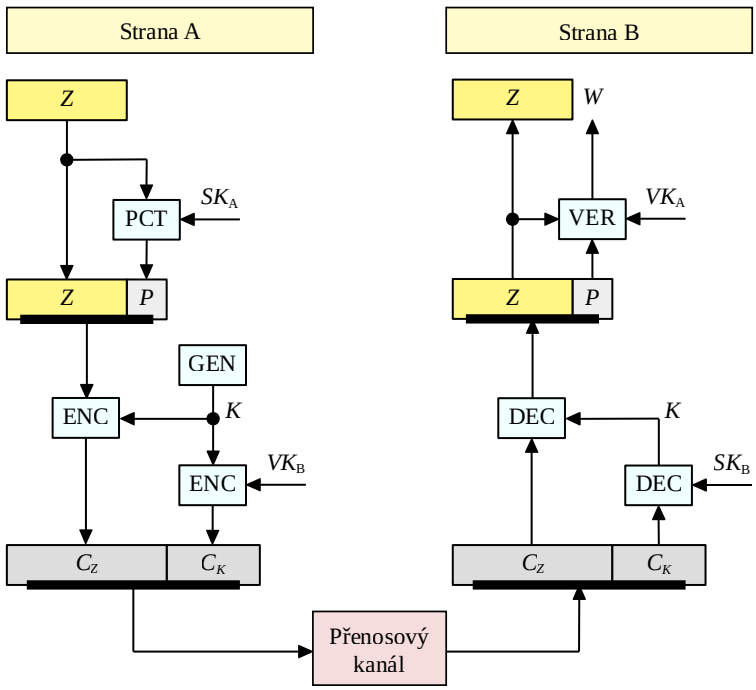
Princip elektronické pošty (viz obr. 2.6) je takový, že odesílatel A ve své aplikaci (tzv. poštovní klient $\mathrm{PK_A}$) vytvoří elektronickou zprávu Z, což je textové sdělení, které je případně doplněno přílohami (např. obrázky). V záhlaví zprávy se kromě jiného nachází poštovní adresa adresáta B. Poštovní klient poté zprávu odešle poštovnímu serveru odesílatele $\mathrm{PS_A}$. Ten podle adresy adresáta B zjistí jeho poštovní server $\mathrm{PS_B}$ a zprávu Z mu předá. Server $\mathrm{PS_B}$ přijatou zprávu uloží do té části svého paměťového úložiště, která je vyčleněna pro uživatele B (tzv. poštovní schránka). Když se uživatel B připojí k serveru $\mathrm{PS_B}$, tak si zprávu ze své schránky nechá poslat na svůj počítač, kde si ji pak pomocí svého poštovního klienta $\mathrm{PK_B}$ přečte.

V původní podobě je systém elektronické pošty bez kryptografického zabezpečení. To znamená, že případný útočník může uživateli B zaslat falešnou zprávu, nebo si útočník může přenášenou zprávu přečíst. Z těchto důvodů vznikla potřeba poštovní zprávy chránit. Ke kryptografickému zabezpečení elektronické pošty vznikly internetové standardy PGP („Pretty Good Privacy“) a S/MIME („Secure/Multipurpose Internet Mail Extensions“) [3]. Jejich princip je stejný – v obou případech se jedná o hybridní kryptosystém v aplikační vrstvě, v němž je zkombinována symetrická a asymetrická šifra spolu s digitálním podpisem. Symetrická šifra přitom slouží k utajení obsahu přenášené zprávy, asymetrická šifra zajišťuje transport klíče pro symetrickou šifru a digitální podpis je použit k autentizaci zprávy.



Obrázek 2.6: Princip elektronické pošty

Princip kryptografického zabezpečení elektronické pošty si vysvětlíme podle schématu na obr. 2.7, přičemž si popíšeme variantu, kdy je zpráva jak podepsána, tak i následně zašifrována. V našem příkladu bude odesílatelem zprávy Z strana A a strana B bude adresátem. Z hlediska klíčů předpokládáme, že obě strany mají svoji dvojici soukromý a veřejný klíč. Strana A má dvojici klíčů SK_A a VK_A, které jsou určeny pro autentizaci zpráv od A a strana B má dvojici klíčů SK_B a VK_B, které jsou určeny pro utajování zpráv pro B. Předpokládáme rovněž, že obě strany získaly bezpečným způsobem veřejné klíče protistrany (např. při osobním setkání nebo z certifikátů).



Obrázek 2.7: Schéma zabezpečení elektronické pošty

Postup je následující. Nejprve strana A podepíše zprávu Z pomocí svého soukromého podpisového klíče SK_A. Podpis $P = \mathrm{PCT}(Z, SK_A)$ připojí ke zprávě, a tak získá podepsanou zprávu $Z_P = (Z, P)$. Dále strana A pomocí generátoru nepředvídatelných čísel náhodně zvolí tajný klíč $K = \mathrm{GEN}$. Zprávu Z_P tímto klíčem zašifruje pomocí symetrické šifry do podoby kryptogramu $C_Z = \mathrm{ENC}(Z_P, K)$. Kromě toho, tentokráte již asymetrickou šifrou, zašifruje klíč K, k čemuž využije veřejný klíč adresáta VK_B. Výsledný kryptogram $C_K = \mathrm{ENC}(K, VK_B)$ pak spolu s kryptogramem C_Z odešle adresátovi B.

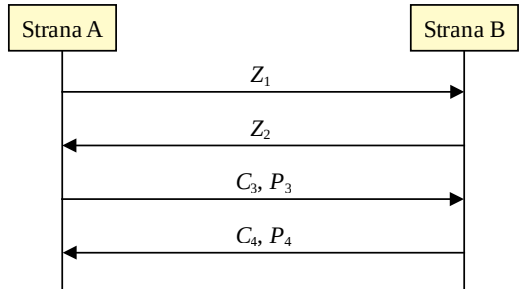
Adresát B nejprve pomocí svého soukromého klíče SK_B dešifruje kryptogram C_K, čímž získá klíč $K = \mathrm{DEC}(C_K, SK_B)$. Pomocí klíče K nyní dešifruje kryptogram C_Z, čímž obdrží podepsanou zprávu $Z_P = \mathrm{DEC}(C_Z, K)$. Nakonec ověří pomocí veřejného klíče VK_A, zda je dešifrovaná zpráva $Z_P = (Z, P)$ autentická. Pokud autentizační indikátor $W = \mathrm{VER}(Z, P, VK_A) = 1$, tak je zpráva považována za autentickou.

2.4.2 Protokol IKE

Protokol IKE („Internet Key Exchange“) je součástí komplexu kryptografického zabezpečení IPsec, což je kryptografické zabezpečení síťového přenosového protokolu IP (podrobněji viz podkapitola 2.6.1). Jedná se o aplikační kryptografický protokol, jehož cílem je ustavit tajné klíče pro protokol AH („Authentication Header“) a protokol ESP („Encapsulating Security Payload“). Kromě ustavení klíčů se v jeho rámci provádí i vzájemná autentizace komunikujících stran, aby byla záruka, že klíče byly skutečně ustaveny se zamýšlenou protistranou. Protokol IKE existuje ve dvou verzích, přičemž my si popíšeme nejnovější verzi, která nese označení IKEv2 [4].

Výměnu zpráv protokolu IKE ilustruje obr. 2.8. Strana A odešle protistraně zprávu Z_1, která sestává z bloků, které označíme NA_{IKE}, $IX_A(\mathrm{B})$, Q_A a N_A. Blok NA_{IKE} je nabídkou kryptografických technik pro protokol IKE. Protistrana B se z této nabídky dozví, jaké kryptografické techniky (tj. jaké typy šifer, pečetí a hešovacích funkcí) strana A dokáže provádět. Blok $IX_A(\mathrm{B})$ obsahuje číselný index, pod nímž si strana A bude ve své databázi vést kryptografické parametry pro spojení se stranou B (podrobněji viz podkapitola k zabezpečení IPsec). Blok Q_A obsahuje Diffie-Hellmanův obraz strany A, tj. $Q_A = \mathrm{DHF}(V, K_A)$, kde V je veřejně známý DH vzor a K_A je tajný klíč strany A. Blok N_A je náhodné číslo, kterým se zajišťuje, že ustavené klíče budou pokaždé jiné.

Strana B zareaguje zprávou Z_2, která v sobě zahrnuje bloky VY_{IKE}, $IX_B(\mathrm{A})$, Q_B a N_B. Z bloku VY_{IKE} se strana A dozví, jaké kryptografické techniky (tj. jaké typy šifer, pečetí a hešovacích funkcí) pro další postup strana B z její nabídky vybrala. Blok $IX_B(\mathrm{A})$ obsahuje číselný index, pod nímž bude strana B ve své databázi vést kryptografické parametry pro spojení se stranou A. Blok Q_B obsahuje DH obraz strany B, tj. $Q_B = \mathrm{DHF}(V, K_B)$, kde V je veřejně známý DH vzor a K_B je tajný klíč strany B. Blok N_B obsahuje náhodné číslo strany B.



Obrázek 2.8: Výměna zpráv v protokolu IKE

Z obrazů Q_A a Q_B jsou nyní obě strany schopny určit tajné semeno $S = \mathrm{DHF}(Q_B, K_A) = \mathrm{DHF}(Q_A, K_B)$. Náhodná čísla obou stran se zřetězí, a tak se získá společný kontext $T = N_A \parallel N_B$, který je použit k odvození bloku $B_{IKE} = \mathrm{ODF}_1(S, T)$, kde ODF_1 je standardem definovaná odvozovací funkce. Blok B_{IKE} je prakticky pseudonáhodným řetězcem bitů o potřebné délce, který závisí na tajném semeni a náhodných číslech. Rozdělením tohoto bloku se získají tajné klíče *KS*, *KP* a *KE*, tj. můžeme psát, že $B_{IKE} = KS \parallel KP \parallel KE$. Klíč *KS* bude později použit k odvození klíčů pro protokol AH, resp. ESP a klíč *KE*, resp. *KP* bude použit k šifrování, resp. pečetění ve zbytku protokolu IKE.

Strana A nyní vytvoří zprávu $Z_3 = (\mathrm{A}, P_A, NA_{IP})$, kde A je identifikátor strany A a P_A je pečeť strany A. Tato pečeť se vypočítává pro řetězec $Z_1 \parallel N_B$, tj. pro řetězec, na jehož hodnotu měla vliv jak strana A (prostřednictvím Z_1), tak i strana B (prostřednictvím N_B). Tím se eliminují různé útoky založené na modifikaci vyměňovaných dat. Pečeť P_A se přitom vypočítá buď pomocí společného pečetícího klíče K_{AB}, nebo pomocí soukromého podepisovacího klíče SK_A strany A. To znamená, že buď $P_A = \mathrm{PCT}(Z_1 \parallel N_B, K_{AB})$, nebo $P_A = \mathrm{PCT}(Z_1 \parallel N_B, SK_A)$. Jaký typ pečetícího kryptosystému bude použit (tj. buď symetrický s klíčem K_{AB}, nebo asymetrický s klíčem SK_A), si obě strany sjednaly v rámci výměny bloků NA_{IKE} a VY_{IKE}. V případě první varianty musí mít obě strany předem ustaven společný pečetící klíč K_{AB} a v případě druhé varianty musí strana A ve zprávě Z_3 doručit straně B také příslušný certifikát $CRT_{CA}(VK_A)$. Blok NA_{IP} je nabídkou kryptografických možností strany A pro protokol AH, resp. ESP.

Dále strana A pomocí odvozeného šifrovacího klíče *KE* zašifruje zprávu Z_3 do podoby kryptogramu $C_3 = \mathrm{ENC}(Z_3, KE)$ a pro tento kryptogram vypočítá pomocí odvozeného pečetícího klíče *KP* hodnotu pečeti $P_3 = \mathrm{PCT}(C_3, KP)$. Dvojici (C_3, P_3) strana A odešle protistraně. Strana B nejprve zkontroluje správnost pečeti P_3. Pokud autentizační indikátor $W = \mathrm{VER}(C_3, P_3, KP) = 1$, tak potom má strana B záruku, že kryptogram C_3 pochází od strany, s níž zahájila protokol IKE. Následně kryptogram C_3 dešifruje, čímž získá zprávu $Z_3 = \mathrm{DEC}(C_3, KE) = (\mathrm{A}, P_A, NA_{IP})$.

Nyní strana B ověří, zda pečeť P_A je platnou pečetí řetězce $Z_1 \parallel N_B$. Pro výstup *W* verifikační funkce musí platit $W = \mathrm{VER}(Z_1 \parallel N_B, P_A, K) = 1$, přičemž ověřovací klíč *K* je v tomto případě

buď společný klíč K_{AB}, nebo veřejný klíč VK_A. Pokud platí, že $W = 1$, tak si tím strana B ověří, že protistranou je skutečně strana s identifikátorem A. Následně strana B z nabídky NA_{IP} vybere kryptografické techniky, které budou použity v kryptografickém protokolu AH, resp. ESP. Tento výběr stanovuje blokem VY_{IP}.

Strana B nyní sestrojí zprávu $Z_4 = (B, P_B, VY_{IP})$, kde B je identifikátor strany B, P_B je pečeť strany B a VY_{IP} je již zmíněný výběr technik pro protokol IPsec. Blok P_B je v tomto případě pečetí řetězce $Z_2 \| N_A$. Vypočítává se analogicky jako pečeť P_A, tj. buď pomocí společného klíče K_{AB}, nebo pomocí soukromého podepisovacího klíče SK_B strany B. Ve druhém z uvedených případů je součástí zprávy i certifikát veřejného klíče strany B. Zpráva Z_4 je poté zašifrována do podoby kryptogramu $C_4 = \mathrm{ENC}(Z_4, KE)$ a pro tento kryptogram se vypočítá pomocí odvozeného pečetícího klíče KP hodnota pečeti $P_4 = \mathrm{PCT}(C_4, KP)$. Dvojici (C_4, P_4) strana B odešle protistraně.

Strana A nejprve zkontroluje správnost pečeti P_4. Pro výstup W verifikační funkce musí platit, že $W = \mathrm{VER}(C_4, P_4, KP) = 1$. V tomto případě má strana A záruku, že kryptogram C_4 pochází od strany, s níž zahájila protokol IKE. Následně strana A přijatý kryptogram C_4 dešifruje, čímž získá zprávu $Z_4 = \mathrm{DEC}(C_4, KE) = (B, P_B, VY_{IP})$. Podobně jako v předešlém případě nyní strana A ověří identitu protistrany. Musí platit, že $W = \mathrm{VER}(Z_2 \| N_A, P_B, K) = 1$, přičemž ověřovací klíč K je buď společný klíč K_{AB}, nebo veřejný klíč VK_B. Pokud platí, že $W = 1$, tak má strana A potvrzeno, že protistranou je skutečně strana s identifikátorem B. Z bloku VY_{IP} se pak strana A dozví kryptografické techniky, které budou použity v protokolech AH, resp. ESP.

Obě strany na závěr protokolu IKE odvodí klíče pro protokol AH, resp. ESP. Z bloku $B_{IKE} = KS \| KE \| KP$ vezmou klíč KS (přesněji semeno) a z něho a ze společného kontextu $T = N_A \| N_B$ odvodí blok $B_{IP} = \mathrm{ODF}_2(KS, T)$, kde ODF_2 je standardem definovaná odvozovací funkce. Získaný blok bitů se rozdělí na příslušné klíče pro protokol AH, resp. ESP. Strana A tyto klíče spolu s indexem $IX_B(A)$ protistrany B uloží do své kryptografické databáze. Ukazatelem (indexem) na uvedené údaje je $IX_A(B)$, tj. index strany A pro klíče ke komunikaci se stranou B. Také strana B si ustavené klíče spolu s indexem protistrany $IX_A(B)$ uloží do své databáze, tentokráte pod vlastním indexem $IX_B(A)$. Tím je cíl protokolu IKE splněn. Později u popisu protokolů AH, resp. ESP uvidíme, že pokud například strana A odešle straně B zabezpečený paket, tak v kryptografickém záhlaví tohoto paketu uvede index $IX_B(A)$. Strana B, která obecně může vést protokol AH, resp. ESP s mnoha dalšími stranami, dokáže podle hodnoty $IX_B(A)$ ve své databázi rychle nalézt odpovídající klíče pro komunikaci se stranou A a s jejich pomocí pak přijatý paket kryptograficky zpracovat.

Obě strany se mohou kdykoliv k protokolu IKE vrátit a dalšími výměnami zpráv mohou mezi sebou ustavovat klíče pro jiné kryptografické algoritmy a režimy, tj. mohou vytvářet nové položky ve své kryptografické databázi. A analogicky mohou tyto položky aktualizovat nebo vymazávat.

2.4.3 Protokol DNSsec

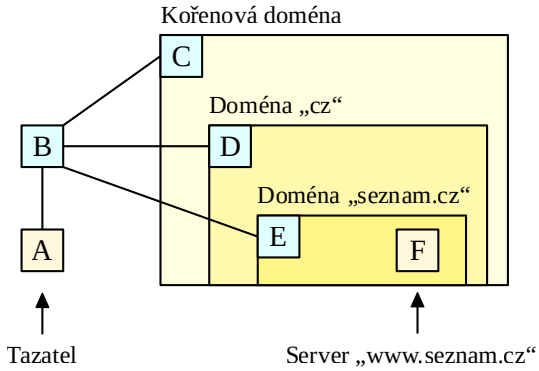
Ke snadno zapamatovatelné identifikaci počítačů v IP sítích byl zaveden systém tzv. doménových jmen („Domain Name System" – DNS). Základem zmíněného systému je tzv. doména, což je určitá množina počítačů. Každou takovou množinu lze libovolně členit na dílčí podmnožiny (tzv. subdomény), přičemž speciálním případem je jednoprvková subdoména, tj. jednotlivý počítač. Popsaný systém domén definuje hierarchii, kdy nejvyšší doménou (tzv. kořenovou doménou) je množina všech počítačů komunikujících přes internet. Kořenové doméně jsou podřízeny její podmnožiny (tzv. domény 1. řádu), těm jsou zase podřízeny jejich subdomény (tzv. domény 2. řádu) atd.

Vzniklá struktura se používá k přidělování jmen doménám a počítačům. Každá doména 1. řádu má přiděleno unikátní jméno D_1. Každé její subdoméně (tj. doméně 2. řádu) se přiděluje jméno D_2, které musí být v rámci dané domény 1. řádu unikátní. Globálně unikátní jméno jakékoliv domény 2. řádu pak vznikne jednoduchým zřetězením jmen D_2 a D_1, přičemž jako oddělovač zde slouží tečka. Analogicky se postupuje u domén 3. a vyšších řádů. Jako ilustrativní příklad nám poslouží počítač se jménem „www.seznam.cz". Podle zmiňovaných pravidel náš počítač náleží do domény 1. řádu, která má jméno „cz", tj. D_1 = „cz". V rámci domény „cz" (doména pro Českou republiku) patří náš počítač do domény s lokálně unikátním jménem D_2 = „seznam", takže globálně unikátní jméno této domény 2. řádu je $D_2.D_1$ = „seznam.cz". V uvedené doméně je náš počítač pojmenován D_3 = „www" (časté jméno pro webové servery), a tak jeho globálně unikátní jméno je $D_3.D_2.D_1$ = „www.seznam.cz".

Uvedli jsme si, jak počítače dostávají v rámci systému DNS své jméno. Protože však k vybudování IP spojení k cílovému počítači je nutná znalost jeho IP adresy, tak pro překlad doménových jmen na IP adresy byl vytvořen stejnojmenný distribuovaný databázový systém. Fungování systému DNS je založeno na tom, že v každé víceprvkové doméně existuje DNS server, který zná jména a IP adresy všech jednotlivých počítačů své domény (svých jednoprvkových domén) a zná jména a IP adresy DNS serverů v podřízených subdoménách. Jak probíhá překlad jména na IP adresu, si vysvětlíme na příkladu podle obr. 2.9. Na obrázku máme webový server F, který má jméno „www.seznam.cz". Jak jsme si již uvedli, jedná se o jednotlivý počítač, a tak jeho jméno a IP adresa je známa DNS serveru E, který má na starosti překlady v doméně 2. řádu „seznam.cz". Jméno a IP adresa DNS serveru E je pak známa DNS serveru D, který zajišťuje překlady v doméně 1. řádu „cz". A nakonec jméno a IP adresu serveru D zná DNS server C, který provádí překlady v kořenové doméně.

Dejme tomu, že uživatel počítače A napsal do adresního řádku webového prohlížeče jméno „www.seznam.cz". Počítač A (nazveme jej tazatelem) je nastaven tak, aby požádal o překlad tohoto jména tzv. lokální DNS server, který máme označen písmenem B. Obvykle se jedná o server firemní sítě, nebo o server poskytovatele internetu. Tyto servery znají IP adresu kořenového serveru C a s žádostí o překlad se obracejí nejprve na něj. Server C má přehled o DNS serverech všech domén 1. řádu, takže serveru B odpoví IP adresou serveru D, který má na starosti doménu „cz".

Server B se poté obrátí se svým dotazem na odkazovaný server D. Ten má přehled o DNS serverech všech svých domén 2. řádu, takže serveru B zašle IP adresu serveru E, který má na starosti doménu „seznam.cz“. Server B se nyní se svým dotazem obrátí na server E, který mu konečně poskytne IP adresu serveru F. Server B tuto adresu předá uživatelskému počítači A, který pak může navázat IP spojení s F.

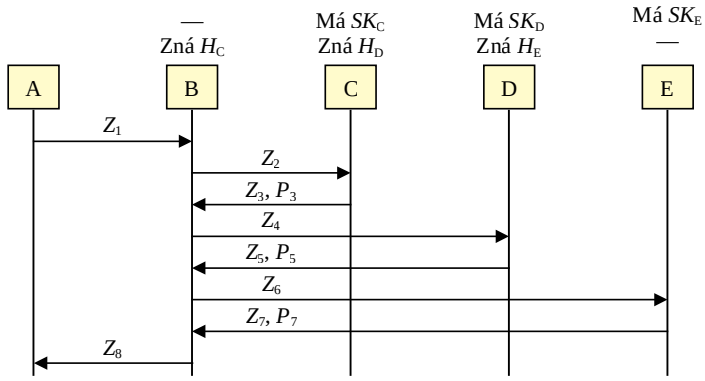


Obrázek 2.9: Příklad systému DNS

Výhodou systému DNS je snadná rozšiřitelnost. Novou subdoménu stačí nahlásit pouze správci nadřízené domény, který ji zaregistruje v DNS serveru své domény. Registrace v jiných DNS serverech není zapotřebí. Slabinou systému DNS je, že útočník může lokálnímu DNS serveru B zasílat na jeho dotazy podvržené odpovědi. Tazatel se pak může například namísto serveru své banky připojit k počítači útočníka, který se bude tvářit jako server falšované banky, a tam zadat a tím prozradit své přihlašovací údaje. K eliminaci uvedené hrozby vznikl protokol DNSsec.

Protokol DNSsec („Domain Name System Security Extensions“) [5] umožňuje lokálním DNS serverům ověřovat pravost odpovědí od ostatních DNS serverů. Prakticky se jedná o kryptografický protokol v aplikační vrstvě, který je založen na podepisování zpráv. Fungování protokolu si vysvětlíme podle obr. 2.10.

Předpokladem je, že doménové servery DNS (v našem příkladu servery C, D a E) si vygenerují své klíče podpisovacího kryptosystému. Soukromé klíče serverů C, D, resp. E označíme SK_C, SK_D, resp. SK_E a jejich veřejné klíče označíme VK_C, VK_D, resp. VK_E. K ověřování veřejných klíčů se v DNSsec zpravidla nepoužívají certifikáty, ale jejich bezpečně doručené heše $H_X = \mathrm{HSF}(VK_X)$. Heš H_C vloží příslušní správci do svých lokálních DNS serverů ručně (v našem příkladu se jedná o server B). Heš H_D, resp. H_E předá příslušný správce serveru bezpečným způsobem správci nadřízeného serveru C, resp. D.



Obrázek 2.10: Výměna zpráv v protokolu DNSsec

Označme si IP adresy všech zúčastněných počítačů jako IP_A až IP_F. Počítač A (Tazatel) odešle lokálnímu DNS serveru (počítač B) zprávu Z_1 s žádostí o překlad jména „www.seznam.cz“. Server B následně odešle kořenovému serveru C zprávu Z_2 s dotazem „Jaká je IP adresa počítače www.seznam.cz?“. Server C odpoví dvojicí (Z_3, P_3), kde Z_3 je zprávou („Zeptej se počítače s IP_D“, H_D, VK_C) a P_3 je podpisem této zprávy, tj. $P_3 = PCT(Z_3, SK_C)$. Jak již víme, hodnota H_D je hešem veřejného klíče odkazovaného serveru D, tj. $H_D = HSF(VK_D)$ a VK_C je veřejný klíč kořenového serveru. Server B nejprve ověří správnost zaslaného klíče VK_C. Musí platit, že $HSF(VK_C)$ je roven hodnotě H_C, která byla správcem serveru uložena do paměti serveru B. Jestliže je uvedený test úspěšný, tak je klíč VK_C použit k ověření podpisu P_3. Pokud pak platí, že autentizační indikátor $W = VER(Z_3, P_3, VK_C) = 1$, tak zprávu Z_3 skutečně zaslal kořenový server C. Lokální server B tak důvěryhodně získal informaci, že svůj dotaz má směrovat na server s adresou IP_D a dozvěděl se, že heš veřejného klíče odkazovaného serveru má mít hodnotu H_D. Pomocí tohoto heše server B za chvíli ověří pravost odpovědi serveru D.

Server B poté zašle serveru D zprávu Z_4 s dotazem „Jaká je IP adresa počítače www.seznam.cz?“. Server D odpoví dvojicí Z_5, P_5, kde Z_5 je zprávou („Zeptej se počítače s IP_E“, H_E, VK_D) a P_5 je podpisem této zprávy, tj. $P_5 = PCT(Z_5, SK_D)$. Hodnota H_E je hešem veřejného klíče odkazovaného serveru E, tj. $H_E = HSF(VK_E)$. Počítač B nejprve ověří správnost zaslaného klíče VK_D. Musí platit, že $HSF(VK_D)$ je roven hodnotě H_D, kterou straně B zaslal kořenový server C. Pokud je tento test úspěšný, tak klíč VK_D použije k ověření podpisu P_5, přičemž popis ověřování již vynecháme, protože se provádí analogicky jako u P_3. Lokální server B tak důvěryhodně získal informaci, že svůj dotaz má nyní směrovat na server s adresou IP_E a dozvěděl se, že heš veřejného klíče odkazovaného serveru má hodnotu H_E.

Server B nyní odešle serveru E zprávu Z_6 opět s dotazem „Jaká je IP adresa počítače www.seznam.cz?“. Server E odpoví dvojicí Z_7, P_7, kde Z_7 je zprávou („Má adresu IP_F“, VK_E) a P_7 je podpisem této

zprávy, tj. $P_7 = \mathrm{PCT}(Z_7, SK_E)$. Počítač B nejprve ověří správnost zaslaného klíče VK_E. Musí platit, že $\mathrm{HSF}(VK_E)$ je roven hodnotě H_E, kterou straně B zaslal DNS server D. Pokud je tento test úspěšný, tak klíč VK_E použije k ověření podpisu P_7, přičemž i zde popis ověřování vynecháme, neboť se opět provádí obdobně jako u P_3. Lokální server B tak konečně získal informaci, že počítač se jménem „www.seznam.cz" má IP adresu IP_F. Toto zjištění předá ve zprávě Z_8 tazateli A, který pak může iniciovat vytvoření IP spojení k serveru F.

Bezpečnost protokolu DNSsec spočívá zejména v bezpečném získání hešů ověřovacích veřejných klíčů. V souvislosti s bezpečností si můžeme ještě povšimnout, že výměna zpráv mezi tazatelem A a lokálním DNS serverem B není kryptograficky chráněna. Spoléhá se na to, že předpokládaní útočníci nemají možnost výměnu zpráv mezi A a B ovlivňovat.

2.5 Kryptografie v transportní vrstvě

Asi nejznámějšími a nejvíce používanými kryptografickými protokoly v transportní vrstvě jsou protokol TLS a protokol SSH. Zde si je vysvětlíme podrobněji.

2.5.1 Protokol TLS

Protokol TLS („Transport Layer Security") [6] je protokol pro kryptografické zabezpečení přenosu zpráv aplikačních protokolů, které využívají transportní protokol TCP. Z tohoto hlediska je možné jej chápat jako kryptografickou nadstavbu protokolu TCP, a tak jej zařadit do transportní vrstvy. Protokol se používá k ochraně komunikace typu klient – server, tj. komunikace, kde specializované počítače (tzv. servery) poskytují počítačům uživatelů (tzv. klientům) různé služby. Příkladem jsou webové servery, které uživatelům nabízejí možnost prohlížení webových stránek se zprávami, nabídkami obchodů apod. Protokol TLS patří k nejrozšířenějším kryptografickým protokolům IP sítí.

Protokol TLS je nástupcem protokolu SSL („Secure Sockets Layer"), který pro svůj webový prohlížeč vyvinula v roce 1994 firma Netscape. Protokol SSL má tři verze (1.0, 2.0, 3.0), z nichž však už ani jedna není považována za bezpečnou. Další vývoj zmiňovaného protokolu převzala internetová standardizační organizace IETF („Internet Engineering Task Force"), která jej vyvíjí pod označením TLS. Protokol TLS existuje ve verzích 1.0, 1.1, 1.2 a 1.3, přičemž se doporučuje využívat jen poslední dvě verze. My se v dalším seznámíme s popisem nejnovější verze, tj. verze 1.3.

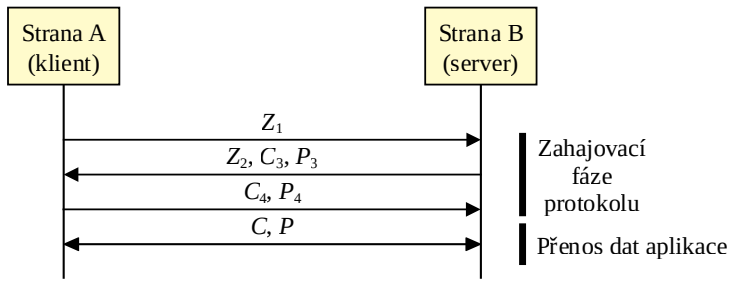
Každá z verzí protokolu TLS zajišťuje:

- ustavení klíčů,
- autentizaci komunikujících stran a
- autentičnost a důvěrnost přenášených dat.

Ustavení klíčů se ve verzi TLS 1.3 obvykle uskutečňuje pomocí DH protokolu, případně technikou předem sjednaného klíče („Pre-Shared Key" – PSK). My si vysvětlíme nejvíce používanou variantu, která je založena na DH protokolu. V tomto případě je autentizace komunikujících stran založena na digitálních certifikátech podpisových klíčů, avšak v praxi se certifikátem autentizuje zpravidla jen server. Případná autentizace klienta se pak řeší až v rámci aplikačního protokolu (typicky pomocí hesla). Autentičnost a důvěrnost aplikačních dat se zajišťuje pomocí symetrických integrovaných kryptosystémů, přičemž ke zjednodušení výkladu nebudeme autentizaci záhlaví datových jednotek TLS protokolu popisovat.

Protokol (viz obr. 2.11) zahajuje klient, tj. strana A. Ta nejprve vygeneruje náhodný tajný klíč K_A a vypočítá DH obraz $Q_A = \mathrm{DHF}(V, K_A)$, kde V je veřejně známý DH vzor. Následně protistraně zašle zprávu $Z_1 = (NA, N_A, Q_A)$, kde NA je nabídka kryptografických možností klienta, tj. seznam typů šifer, pečetí, hešovacích funkcí a podpisů, které strana A dokáže provádět. Blok N_A je náhodně vygenerované číslo, které se později uplatní při odvozování klíčů a číslo Q_A je již zmíněný DH obraz.

Server (tj. strana B) po přijetí zprávy Z_1 vygeneruje svůj náhodný tajný klíč K_B a vypočítá pro něj DH obraz $Q_B = \mathrm{DHF}(V, K_B)$. Rovněž tak vygeneruje náhodné číslo N_B a vytvoří zprávu $Z_2 = (VY, N_B, Q_B)$, kde VY je výběr serveru z nabídky NA. Tím se strana A později dozví, jaké kryptografické algoritmy budou použity v další komunikaci. Server dále vypočítá tajné semeno („Handshake Secret") $S = \mathrm{DHF}(Q_A, K_B)$, z něhož se pak pomocí funkce ODF_1 odvodí zahajovací klíč $KZ = \mathrm{ODF}_1(S, T_1)$, přičemž kontext T_1 je hešem zpráv Z_1 a Z_2, tj. $T_1 = \mathrm{HSF}(Z_1 \| Z_2)$. Zahajovací klíč bude použit k šifrování a pečetění všech dalších zpráv až do konce zahajovací fáze protokolu.



Obrázek 2.11: Výměna zpráv v protokolu TLS verze 1.3

Poznamenáváme, že proces odvozování klíčů je ve standardu TLS 1.3 dosti komplikovaný, a tak si jeho popis zjednodušíme tím, že zavedeme několik funkcí ODF, které odlišíme indexy. Server B pak ještě kromě zahajovacího klíče odvodí signalizační klíč KS. Ten se použije k vytvoření bloku, jímž se signalizuje připravenost k přenosu dat aplikace. Pro tento klíč platí, že $KS = \mathrm{ODF}_2(S, T_1)$, kde ODF_2 je další odvozovací funkce standardu.

Server dále sestaví zprávu $Z_3 = (CR_B, SG_B, PS_B)$. Blok CR_B je certifikátem ověřovacího klíče VK_B serveru, tj. $CR_B = CRT_{CA}(VK_B)$. Blok SG_B je podpisem heše $H_1 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B)$, tj. $SG_B = \mathrm{PCT}(H_1, SK_B)$, kde SK_B je soukromý podepisovací klíč serveru B. Tímto podpisem server B prokáže straně A vlastnictví soukromého SK_B, který je komplementární k VK_B z certifikátu. Server B se jím tak bude autentizovat straně A. Všimněme si, že zmiňovaný podpis závisí na heši zpráv Z_1, Z_2 a bloku CR_B. A protože unikáty N_A a N_B ze zpráv Z_1 a Z_2 jsou pro každý běh protokolu jedinečné, tak podpis SG_B je pokaždé platný jen pro daný běh protokolu. Případný útočník nemůže tyto podpisy použít k oklamání klienta v jiných bězích protokolu.

Signální pečeť PS_B signalizuje konec zahajovací fáze protokolu TLS ze strany serveru a zároveň i jeho připravenost k přenosu dat aplikace. Jedná se o pečeť hodnoty heše ze všech předchozích zpráv i bloků, která je vytvořena pomocí signalizačního klíče *KS*, tj. $PS_B = \mathrm{PCT}(H_2, KS)$, kde $H_2 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B \parallel SG_B)$. Strana A si později kontrolou této pečeti ověří, že server ukončil zahajovací fázi protokolu a je schopen přenosu dat aplikace. Také tím získá záruku, že všechny předešlé zprávy a bloky (tj. vstupy pro H_2) nikdo během jejich přenosu nemodifikoval. Ke zprávě Z_3 se pomocí funkce AEF integrovaného kryptografického systému a klíče *KZ* určí dvojice $(C_3, P_3) = \mathrm{AEF}(Z_3, KZ)$, kde C_3 je kryptogram a P_3 je pečeť. Trojici (Z_2, C_3, P_3) pak server odešle protistraně A.

Strana A nejprve z přijaté zprávy $Z_2 = (VY, N_B, Q_B)$ zjistí podle údajů uvedených v bloku výběru *VY*, jaké kryptografické algoritmy má v dalších výpočtech použít. Klient A posléze vypočítá hodnotu tajného semene $S = \mathrm{DHF}(Q_B, K_A)$ a sestrojí kontext $T_1 = \mathrm{HSF}(Z_1 \parallel Z_2)$. Z nich pak odvodí zahajovací klíč $KZ = \mathrm{ODF}_1(S, T_1)$ a signalizační klíč $KS = \mathrm{ODF}_2(S, T_1)$. Následně pomocí inverzní funkce AEI a klíče *KZ* dešifruje a ověří dvojici (C_3, P_3), tj. získá $\mathrm{AEI}(C_3, P_3, KZ) = (Z_3, W)$. Pokud autentizační indikátor $W = 1$, tak je dešifrovaná zpráva $Z_3 = (CR_B, SG_B, PS_B)$ považována za autentickou. Blok CR_B je certifikátem ověřovacího klíče VK_B serveru, tj. $CR_B = CRT_{CA}(VK_B)$. Klient A pomocí veřejného klíče VK_{CA} příslušné certifikační autority CA přijatý certifikát nejprve ověří. Získá tak záruku, že autentičnost zpráv od serveru B lze ověřovat klíčem VK_B. Tuto skutečnost využije při ověřování podpisu SG_B. Musí platit, že $\mathrm{PCT}(H_1, SG_B, VK_B) = 1$, kde $H_1 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B)$. V takovémto případě si klient A ověřil identitu serveru B. Jenom server B může znát soukromý podepisovací klíč SK_B, který je komplementární k veřejnému klíči VK_B.

Následně ještě klient A ověří signální pečeť PS_B. Pokud platí, že $\mathrm{VER}(H_2, PS_B, KS) = 1$, kde $H_2 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B \parallel SG_B)$, tak strana A má záruku, že veškerá doposud přenesená data, nebyla nikým modifikována. Zároveň se také dozví, že server ukončil zahajovací fázi protokolu a je připraven na přenos dat aplikace. Nyní musí ukončit zahajovací fázi i strana A. K tomu vypočítá blok PS_A, který je analogem PS_B ze strany serveru. I v tomto případě se jedná o pečeť heše ze všech předchozích zpráv i bloků, která je vytvořena pomocí signalizačního klíče *KS*, tj. $PS_A = \mathrm{PCT}(H_3, KS)$, kde $H_3 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B \parallel SG_B \parallel PS_B)$. Strana B si později kontrolou této pečeti ověří, že A se dopracovala k téže hodnotě semena *S* a rovněž se také dozví, že i klient ukončil zahajovací fázi protokolu a je připraven pro přenos dat aplikace.

Blok PS_A je jediným blokem poslední zprávy Z_4 zahajovacího protokolu, tj. $Z_4 = PS_A$. Tato zpráva je zašifrována a zapečetěna pomocí klíče KZ, čímž vznikne dvojice $(C_4, P_4) = \mathrm{AEF}(Z_4, KZ)$, kde C_4 je kryptogram a P_4 je pečeť. Uvedená dvojice je odeslána serveru B, který ji dešifruje a ověří, tj. vypočítá $\mathrm{AEI}(C_4, P_4, KZ) = (Z_4, W)$. Pokud $W = 1$, tak server získanou zprávu Z_4 akceptuje. Ze zprávy vyjme blok PS_A a ten ověří. Jestliže $\mathrm{VER}(H_3, PS_A, KS) = 1$, kde $H_3 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B \parallel SG_B \parallel PS_B)$, tak si rovněž server B ověří, že nikdo žádnou zprávu či blok zahajovacího protokolu nepozměnil a zároveň se mu dostane potvrzení, že klient A je připraven k přenosu aplikačních dat.

K šifrování a pečetění aplikačních dat potřebují obě strany hlavní klíč KH. Ten získají tak, že $KH = \mathrm{ODF}_3(S, T_2)$, kde ODF_3 je stanovený typ odvozovací funkce, kontext $T_2 = H_3 = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel CR_B \parallel SG_B \parallel PS_B)$ a S je tajné semeno. Od tohoto okamžiku lze přenášet zprávy Z aplikačního protokolu, jako je například protokol HTTP. Zpráva Z je pomocí sjednaného integrovaného kryptosystému zašifrována do podoby kryptogramu C a opatřena pečetí P, tj. vypočítá se $(C, P) = \mathrm{AEF}(Z, KH)$. Dvojice (C, P) se odešle protistraně. Protistrana z ní vypočítá $(Z, W) = \mathrm{AEI}(C, P, KH)$. Pokud autentizační indikátor $W = 1$, tak je zpráva Z předána cílové aplikaci.

Seznámili jsme se, jak funguje obvyklá varianta protokolu TLS ve verzi 1.3. Z popisu je zřejmé, že obě strany si v průběhu zahajovací fáze ustaví klíče pro šifrování a pečetění aplikačních dat a viděli jsme, že se server B pomocí svého podpisu autentizoval vůči klientovi A. Případná autentizace klienta A vůči serveru B se v praxi provádí v rámci aplikačního protokolu, tj. mimo rámec protokolu TLS. Klient například na webové stránce serveru vloží své jméno a heslo. V důsledku autentizace serveru má klient záruku, že heslo zasílá oprávněné straně B a také ví, že heslo bude přenášeno v zašifrované podobě, takže případný útočník je nebude schopen zjistit.

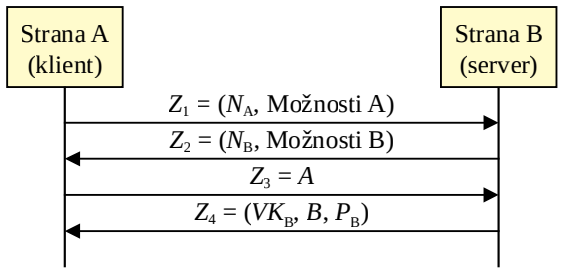
2.5.2 Protokol SSH

Protokol SSH („Secure Shell“) je protokol, který umožňuje kryptograficky zabezpečit různé aplikační protokoly, a proto jej zahrneme do transportní vrstvy. Nejvíce se používá ke vzdálené správě počítačů. Architektura tohoto protokolu je definována v RFC 4251 [7], kryptografické zabezpečení (tj. ustavení klíčů, autentizace serveru, důvěrnost a autentičnost přenášených dat) stanovuje RFC 4253 [8], autentizaci uživatelů definuje RFC 4252 [9] a multiplexování komunikace několika aplikačních protokolů do jediného SSH spojení popisuje RFC 4254 [10].

V rámci protokolu SSH se nejprve provede fáze ustavení klíčů, která je založena na DH protokolu. Dále se pokračuje autentizací serveru, v níž se fakticky využívá digitální podpis. Poté proběhne fáze autentizace klienta, která je obvykle založena buď na heslu, nebo na digitálním podpisu. Následně se uskuteční šifrovaný a autentizovaný přenos dat. Autentičnost veřejných klíčů stran je řešena buď pomocí SSH certifikátů, nebo bezpečným uložením veřejných klíčů v lokální databázi stanice.

Fáze ustavení klíčů a autentizace serveru probíhá následovně (viz obr. 2.12). Klient A odešle serveru zprávu Z_1, která obsahuje náhodné číslo N_A klienta a seznamy algoritmů, které klient

dokáže provádět (na obrázku Možnosti A). Seznamy jsou uspořádány po kategoriích od DH algoritmů, přes podpisové algoritmy serveru, šifrovací algoritmy a pečetící algoritmy. Položky ve všech uvedených seznamech jsou uspořádány podle klesající preference klienta. Server B odpoví zprávou Z_2, která obsahuje náhodné číslo N_B serveru a seznamy algoritmů, které dokáže provádět server. Tyto seznamy jsou uspořádány tentokráte podle preferencí serveru. Z vyměněných seznamů si strany pro každou kategorii algoritmů provedou průnik, tj. zjistí algoritmy, které dokáží provádět obě. Pokud se v dané kategorii shodly na více možnostech, tak se z nich volí algoritmus, který klient preferoval nejvýše.



Obrázek 2.12: Fáze ustavení klíčů a autentizace serveru v protokolu SSH

Po ustavení algoritmů následuje fáze sjednání klíčů. Klient A nejprve zvolí náhodný DH klíč a a k němu pro stanovený vzor V vypočítá jeho DH obraz $A = \mathrm{DHF}(V, a)$. Serveru pak odešle zprávu $Z_3 = A$. Server zvolí svůj náhodný DH klíč b a k němu pro stanovený vzor V vypočítá DH obraz $B = \mathrm{DHF}(V, b)$. Rovněž tak určí tajné semeno $S = \mathrm{DHF}(A, b)$. Nyní může určit heš $h = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel VK_B \parallel A \parallel B \parallel S)$, kde VK_B je veřejný klíč serveru. Tento heš je v důsledku unikátů N_A a N_B ve zprávách Z_1 a Z_2 unikátním číslem, jímž se individualizuje dané SSH spojení. Hodnota h je tajná, nepřenáší se a nazývá se identifikátor SSH spojení. Server identifikátor h následně podepíše, tj. vypočítá podpis $P_B = \mathrm{PCT}(h, SK_B)$, kde SK_B je soukromý klíč serveru. Nakonec server odešle klientovi zprávu $Z_4 = (VK_B, B, P_B)$.

Klient pomocí certifikátu či lokální databáze nejprve ověří správnost veřejného klíče VK_B. Poté vypočítá hodnotu semene $S = \mathrm{DHF}(B, a)$, takže pak může vypočítat i tajný identifikátor spojení $h = \mathrm{HSF}(Z_1 \parallel Z_2 \parallel VK_B \parallel A \parallel B \parallel S)$. Pro tuto hodnotu ověří správnost podpisu P_B. Pokud platí, že autentizační indikátor $W = \mathrm{VER}(h, P_B, VK_B) = 1$, tak je identita serveru považována za prokázanou.

Obě strany z hodnoty semene S a identifikátoru h odvodí pomocí stanovené odvozovací funkce ODF klíč pro šifrování KE a pro pečetění KP. Platí, že $(KE \parallel KP) = \mathrm{ODF}(S \parallel h, T)$, kde T je standardem stanovený kontext. Od této chvíle je veškerá další komunikace mezi oběma stranami šifrována a pečetěna (viz dále).

Po fázi ustavení klíčů a autentizaci serveru je zahájena fáze autentizace klienta. Nejčastěji se v praxi používá autentizace heslem nebo autentizace veřejným klíčem klienta. V případě autentizace heslem zašle klient zprávu, která obsahuje identifikátor klienta, identifikátor požadované služby a klientovo heslo PSW_A. Server si ve své databázi ověří, zda klient A uvedl správné heslo a v kladném případě mu odpoví zprávou, že autentizace byla úspěšná a požadovanou službu (například vzdálenou správu) mu poskytne.

V případě autentizace veřejným klíčem zašle klient zprávu, která opět obsahuje identifikátor klienta, identifikátor požadované služby, veřejný klíč klienta VK_A a podpis $P_A = \mathrm{PCT}(h, SK_A)$, kde h je identifikátor spojení. Server z certifikátu nebo lokální databáze zkontroluje správnost veřejného klíče VK_A a ověří podpis klienta. Pokud autentizační indikátor $W = \mathrm{VER}(h, P_A, VK_A) = 1$, tak je identita klienta považována za prokázanou. Server poté klientovi zašle zprávu, že autentizace byla úspěšná a požadovanou službu mu poskytne.

Data pro zabezpečený přenos jsou v rámci protokolu SSH dělena na datové jednotky, přičemž každé datové jednotce D je přiděleno pořadové číslo SN. Datová jednotka je poté zašifrována do podoby kryptogramu $C = \mathrm{ENC}(D, KE)$ a opatřena pečetí $P = \mathrm{PCT}(SN \parallel D, KP)$. Protistraně je zaslána dvojice (C, P). Pořadové číslo SN se nepřenáší, protože protokol SSH využívá k přenosu protokol TCP, který garantuje, že data jsou přenesena všechna a ve správném pořadí. Protistrana tak může každé přijaté dvojici (C, P) přiřadit správné pořadové číslo SN. Příjemce nejprve dešifruje kryptogram, tj. vypočítá datovou jednotku $D = \mathrm{DEC}(C, KE)$ a poté zkontroluje pečeť P. Pokud $W = \mathrm{VER}(SN \parallel D, P, KP) = 1$, tak je autentičnost datové jednotky D ověřena a přijatá data jsou předána k dalšímu zpracování.

2.6 Kryptografie v síťové vrstvě

Z hlediska kryptografického zabezpečení v síťové vrstvě zcela dominuje komplex IPsec. Označení komplex zde na rozdíl od označení protokol použijeme proto, že IPsec není jednotlivý protokol, ale jak dále uvidíme, jedná se o ucelený soubor protokolů pro kryptografické zabezpečení přenosů spojených s protokolem IP. Dále si v této podkapitole popíšeme i kryptografické zabezpečení anonymizující sítě Tor.

2.6.1 Komplex IPsec

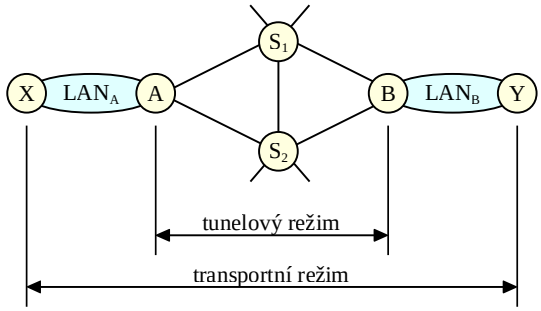
Komplex IPsec je ucelený soubor standardů pro kryptografické zabezpečení síťového přenosového protokolu IP („Internet Protocol“). Kromě kryptografického zabezpečení samotného IP protokolu se výše uvedenými standardy automaticky zajišťuje i bezpečnost všech protokolů, jejichž datové jednotky jsou v tělech IP paketů přenášeny. V praxi se zejména jedná o transportní protokoly TCP i UDP a pak i o všechny aplikační protokoly, které služby protokolů TCP a UDP využívají.

Standardy kryptografického komplexu IPsec definují:

- Bezpečnostní architekturu síťové vrstvy [11].
- Protokol IKE [4] pro ustavení klíčů.
- Protokoly AH [12] a ESP [13], které jsou kryptografickými rozšířeními protokolu IP.
- Různé typy šifer, pečetí, nebo jejich kombinací, jimiž jsou protokoly AH, ESP a IKE konkrétně implementovány.

Bezpečnostní architekturou síťové vrstvy, ani konkrétními šiframi, pečetěmi a jejich kombinacemi se zde zabývat nebudeme. A protože protokol IKE jsme si již vysvětlili, tak nám zbývá si popsat jen protokoly AH a ESP. Již bylo řečeno, že protokol AH („Authentication Header“) i protokol ESP („Encapsulating Security Payload“) jsou prakticky kryptografickými rozšířeními přenosového protokolu IP. Protokol AH je určen k autentizaci dat z těla paketu a k autentizaci těch služebních dat ze záhlaví paketu, která se během přenosu sítí nemění. Naproti tomu protokol ESP slouží především k utajení a případně také k autentizaci dat přenášených v těle paketu. Autentizaci služebních dat ze záhlaví paketu nezajišťuje.

Protokol AH i ESP mohou být provozovány v transportním, nebo v tunelovém režimu. Uvedené režimy si vysvětlíme podle obr. 2.13, kde vidíme dvě lokální sítě LAN_A a LAN_B. Z první sítě máme vyobrazen uživatelský počítač (stanice X) a hraniční směrovač A. Ve druhé síti vidíme stanici Y a hraniční směrovač B. Oba směrovače spolu s dalšími směrovači S_1 a S_2 tvoří segment sítě internetu.



Obrázek 2.13: Tunelový a transportní režim

V případě transportního režimu protokol AH, resp. ESP vykonávají obě koncové stanice X a Y. Výhodou tohoto režimu je, že přenášené pakety jsou zabezpečeny po celé trase od stanice X až ke stanici Y. Pokud si však uvědomíme, že v obou lokálních sítích mohou být až desítky stanic, tak se potom kryptografický provoz a správa klíčů stávají značně komplikovanými. Z tohoto důvodu se v praxi vesměs setkáváme s tunelovým provozem, kdy jsou pakety zabezpečeny pouze na jejich

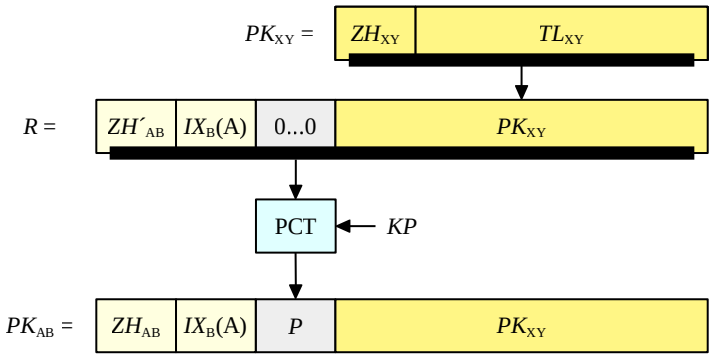
cestě internetem. To znamená, že stranami protokolu ESP, resp. AH jsou v uvedeném případě směrovače A a B. Výhodou tunelového provozu je, že zúčastněných stran je mnohem méně (prakticky pouze hraniční směrovače lokálních sítí), a tak je provoz i správa tohoto systému mnohem jednodušší. Nevýhodou tunelového provozu je, že pakety přenášené v lokálních sítích LAN_A a LAN_B nejsou protokoly AH, resp. ESP chráněny. V tomto případě se spoléhá na to, že případní útočníci nemají v důsledku jiných typů bezpečnostních opatření k lokálním sítím přístup. Tunelový režim využívají zejména velké organizace, které mají své sítě umístěny v různých lokalitách (např. firemní centrála je v jednom státě a její pobočky se nacházejí v jiných státech). Počítačové sítě vzniklé propojením lokálních sítí dané organizace pomocí IPsec tunelů se obvykle nazývají virtuální privátní sítě („Virtual Private Network" – VPN).

Nyní si popíšeme protokoly AH a ESP podrobněji, přičemž se omezíme na tunelový režim, který je v praxi převažující. Kryptografické klíče mezi směrovači A a B se ustavují buď pomocí kurýra, nebo obvykleji pomocí protokolu IKE. Tento protokol jsme si již popsali v podkapitole 2.4.2, a tak jen připomínáme, že obě strany na závěr protokolu IKE odvodí blok B = ODF(KS, T), kde ODF je standardem definovaná odvozovací funkce, T je kontext daný navzájem vyměněnými unikáty a hodnota KS je semeno odvozené pomocí DH protokolu v úvodní fázi protokolu IKE. Z bloku B si poté obě strany stanoveným způsobem odvodí potřebné klíče. V případě protokolu AH to je pečetící klíč KP a v případě protokolu ESP se jedná o šifrovací klíč KE a pečetící klíč KP. Kromě ustavení klíčů se v rámci protokolu IKE obě strany také informovaly o hodnotách svých databázových indexů IX_A(B) a IX_B(A) pro dané spojení.

Strana A ustavené klíče spolu s indexem IX_B(A) protistrany B uloží do své kryptografické databáze. Ukazatelem (indexem) na uvedené údaje je IX_A(B), což je index strany A pro provoz se stranou B. Také strana B si ustavené klíče spolu s indexem protistrany IX_A(B) uloží do své databáze, tentokráte pod vlastním indexem IX_B(A). Pokud potom například strana A odešle straně B zapečetěný, resp. zašifrovaný paket, tak v kryptografickém záhlaví tohoto paketu uvede index IX_B(A). Strana B, která obecně může vést kryptografický provoz s mnoha dalšími stranami, dokáže podle hodnoty IX_B(A) ve své databázi rychle nalézt odpovídající klíče pro komunikaci se stranou A a s jejich pomocí pak ověřit pečeť přijatého paketu, resp. paket dešifrovat.

Sestrojení kryptograficky zabezpečeného paketu protokolem AH v tunelovém režimu vidíme na obr. 2.14, přičemž budeme předpokládat, že paket se bude přenášet od stanice X ke stanici Y.

V horní části obrázku vidíme paket PK_{XY}, který stanice X připravila k odeslání pro stanici Y. V záhlaví paketu ZH_{XY} se nacházejí zejména IP adresy obou stanic a v těle paketu TL_{XY} se obvykle nachází datová jednotka transportního protokolu TCP nebo UDP. Stanice X odešle paket do sítě LAN_A, která jej doručí hraničnímu směrovači A. Ten podle cílové IP adresy paketu zjistí, že paket je určen stanici Y, která se nachází v LAN_B, tj. nachází se za směrovačem B. Ve speciální přiřazovací tabulce má směrovač A uvedeno, že kryptografické údaje pro provoz ke směrovači B má zapsány ve své databázi pod indexem IX_A(B). Pod tímto indexem v databázi nalezne pečetící klíč KP a index protistrany IX_B(A).



Obrázek 2.14: Protokol AH v tunelovém režimu

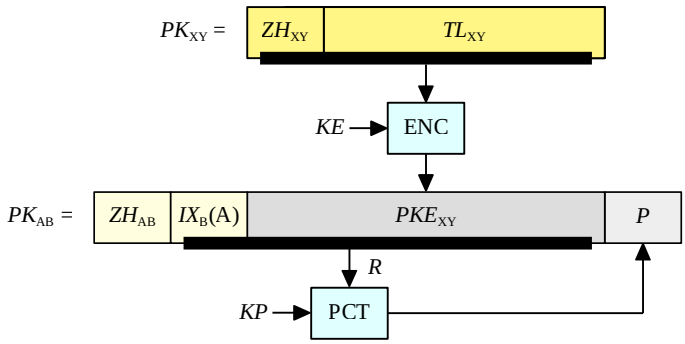
K přenosu paketu PK_{XY} internetovou sítí si směrovač A připraví nový paket PK_{AB} (na obrázku dole), v jehož záhlaví ZH_{AB} se zejména nacházejí IP adresy směrovače A a směrovače B. Pro výpočet autentizační pečeti P si strana A toto záhlaví nejprve upraví do podoby ZH'_{AB} (na obrázku uprostřed). Zmíněná úprava spočívá v tom, že služební data, která se při průchodu internetovou sítí mění, jsou zde nastavena na hodnotu nula. Za upraveným záhlavím ZH'_{AB} směrovač umístí index protistrany IX_B(A), poté řetězec nulových bitů o délce budoucí pečeti P a nakonec paket PK_{XY}. Vznikne tak řetězec $R = ZH'_{AB} \parallel IX_B(A) \parallel (0...0) \parallel PK_{XY}$, pro nějž směrovač A vypočítá pečeť $P = \mathrm{PCT}(R, KP)$. Strana A poté zkompletuje finální paket $PK_{AB} = ZH_{AB} \parallel IX_B(A) \parallel P \parallel PK_{XY}$ a odešle jej protistraně.

Strana B z přijatého paketu vyjme pečeť P, nahradí ji řetězcem nul a nulami rovněž nahradí ta data záhlaví ZH_{AB}, která se průchodem internetovou sítí mění. Získá tak řetězec R'. Pokud nedošlo během přenosu k nežádoucí změně dat v paketu, tak musí platit, že $R' = R$. Strana B nyní vypočítá vlastní hodnotu pečeti $P' = \mathrm{PCT}(R', KP)$. Pokud $P' = P$, tak tím strana B získala záruku, že přijatý paket pochází od strany, která disponuje tajným klíčem KP, tj. pochází od strany A. V takovémto případě vyjme z těla paketu původní paket PK_{XY} a ten odešle do sítě LAN_B. Ta se postará o doručení paketu stanici Y.

Ověření přijatého paketu stranou B jsme záměrně popsali bez použití verifikační funkce VER. Čtenář si tak může udělat lepší představu o tom, jak ve skutečnosti probíhá ověřování pečetí u symetrických autentizačních kryptosystémů. Principem je, že ověřující strana vypočítá z přijaté zprávy vlastní hodnotu pečeti P' a tu porovná s doručenou pečetí P. V případě jejich shody má garantovánu autentičnost zprávy. Pro ověřovací funkci VER je tedy rovnost $P' = P$ kritériem pro nastavení autentizačního indikátoru W na hodnotu 1. V každém jiném případě $W = 0$.

Nyní se věnujme protokolu ESP. Jak již bylo uvedeno, na rozdíl od protokolu AH zajišťuje nejen autentičnost dat v těle paketu, ale i jejich důvěrnost. Na druhou stranu však nechrání autentičnost záhlaví přenášeného paketu. Sestrojení kryptograficky zabezpečeného paketu protokolem ESP

v tunelovém režimu vidíme na obr. 2.15, přičemž opět budeme předpokládat, že paket se přenáší od stanice X ke stanici Y.



Obrázek 2.15: Protokol EPS v tunelovém režimu

V horní části obrázku vidíme paket PK_{XY}, který stanice X připravila k vyslání pro stanici Y. V záhlaví paketu ZH_{XY} se nacházejí opět IP adresy obou stanic a v těle paketu TL_{XY} se nachází datová jednotka transportního protokolu TCP nebo UDP. Stanice X odešle tento paket do sítě LAN_A, která jej doručí hraničnímu směrovači A. Ten podle cílové IP adresy paketu zjistí, že paket je určen stanici Y, která se nachází v LAN_B, tj. nachází se za směrovačem B. Ve speciální přiřazovací tabulce má směrovač A uvedeno, že kryptografické údaje pro provoz ke směrovači B má zapsány ve své databázi pod indexem $IX_A(B)$. Pod tímto indexem v databázi nalezne šifrovací klíč KE, pečetící klíč KP a index protistrany $IX_B(A)$.

V dolní části obrázku vidíme, že směrovač A nejprve vytvoří nové záhlaví ZH_{AB}, které v sobě zejména obsahuje IP adresy směrovačů A a B. Následně zašifruje paket PK_{XY} do podoby PKE_{XY} = ENC(PK_{XY}, KE) a vypočítá pečeť P = PCT(R, KP), kde řetězec $R = IX_B(A) \parallel PKE_{XY}$. Nakonec zkompletuje finální paket $PK_{AB} = ZH_{AB} \parallel IX_B(A) \parallel PKE_{XY} \parallel P$ a ten odešle protistraně.

Strana B z přijatého paketu vezme řetězec $R = IX_B(A) \parallel PKE_{XY}$ a vypočítá svoji hodnotu pečeti $P' = PCT(R, KP)$. Pokud vypočtená pečeť P' je rovna přijaté pečeti P, tak je paket považován za autentický. Strana B následně dešifruje kryptogram PKE_{XY}, tj. vypočítá původní paket PK_{XY} = DEC(PKE_{XY}, KE). Podle cílové IP adresy v dešifrovaném záhlaví ZH_{XY} směrovač B zjistí, že adresátem paketu je stanice Y. Paket odešle do sítě LAN_B, která se již postará o jeho doručení.

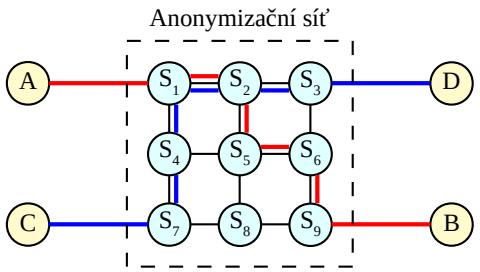
Na závěr této podkapitoly je ještě vhodné zmínit, že z dvojice protokolů AH a ESP v praxi zcela dominuje protokol ESP. Je to z toho důvodu, že k autentičnosti dat navíc zajišťuje i důvěrnost dat. Vzniká proto přirozená otázka, proč vůbec protokol AH vznikl. V devadesátých letech minulého století, kdy se komplex IPsec formoval, některé státy zakazovaly nestátním subjektům šifrování

přenášených dat. Pro tyto subjekty měl protokol AH zajistit alespoň ochranu před útoky na autentičnost přenášených dat.

2.6.2 Anonymizační síť Tor

Anonymizační síť je pojem, který je všeobecně málo známý i poněkud vágní, a proto si jej nejprve přiblížíme. Anonymizační sítí budeme v dalším rozumět přenosovou síť, která zajišťuje anonymitu svých stran. Za anonymitu strany je přitom považován stav, že pokud je dána nějaká strana A, s níž prostřednictvím anonymizační sítě komunikuje strana B, pak by nemělo být možné identitu strany B vypátrat. V této souvislosti je zapotřebí uvést, že anonymita může být jednostranná a oboustranná. Typickým příkladem jednostranné anonymity je situace, kdy uživatel využívá služby poskytované serverem s veřejně známou identitou a před tímto serverem skrývá svoji identitu. Avšak cílovým serverem uživatelů anonymizační sítě mohou být i tzv. anonymní servery, které svoji identitu skrývají stejně jako jejich uživatelé. V takovémto případě se jedná o oboustrannou anonymitu.

Princip anonymizační sítě si vysvětlíme podle obr. 2.16. Vidíme na něm anonymizační síť, která sestává z 9 navzájem propojených směrovačů S_1 až S_9, přičemž k této síti jsou připojeny strany A až D. Na našem obrázku strana A komunikuje se stranou B (přenos zpráv probíhá po červené spojovací cestě) a strana C komunikuje s D (po modré cestě).



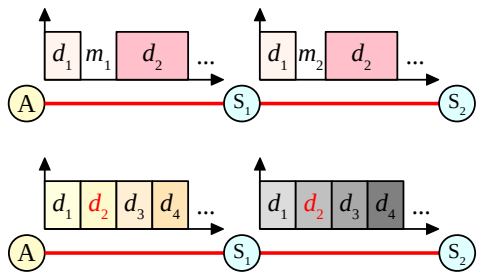
Obrázek 2.16: Anonymizační síť

V anonymizační síti je síťová adresace řešena tak, aby připojené strany nebyly schopny z adres v přenášených datových jednotkách zjistit identitu protistrany. K tomu se obvykle používají techniky substituce zdrojových adres a cílových adres. V případě substituce zdrojových adres se zpravidla nahrazují síťové adresy odesílatelů adresami směrovačů. Příjemce paketu pak není schopen z veřejně přístupných databází síťových IP adres určit identitu skutečného odesílatele. V případě substituce cílových adres je často pro zdrojovou stranu nahrazena identita cílové strany vhodnou přezdívkou. K překladu přezdívky na cílovou adresu pak dochází až v anonymizační síti.

Obecný princip fungování sítě je následující. Iniciátor komunikace (např. strana A) se připojí přes zvolený směrovač (v našem případě přes S_1) k anonymizační síti a tato síť mu pak přes několik dalších směrovačů (podle obrázku přes posloupnost S_1, S_2, S_5, S_6, S_9) zajistí komunikaci s žádanou protistranou B. Směrovače přitom obvykle nahrazují zdrojové adresy v datových jednotkách síťové vrstvy svými adresami. Jak již bylo řečeno, protistrana B často bývá veřejný server a v takovémto případě je pak anonymní pouze strana A vůči straně B. Strana B však může být i anonymní server, kdy je potom anonymita oboustranná. Pro tento případ jsme si již uvedli, že straně A se skutečná identita anonymního serveru B (a s ním i jeho síťová adresa) často prezentuje vhodnou přezdívkou. Provozovatel anonymizační sítě ve všech výše uvedených případech samozřejmě může vědět, koho a s kým jeho síť propojuje. Uživatelé anonymizační sítě však provozovateli sítě důvěřují, že uvedené informace nikomu neposkytne a ani je nezneužije.

Praxe ukázala, že řešit anonymizaci stran jen pomocí síťových adres je nedostatečné. Identitu stran lze totiž vypátrat také analýzou provozu v síti. Pokud například útočník podezřívá, že strana A komunikuje se stranou B, tak toto podezření si útočník může potvrdit monitorováním provozu ve spojích A-S_1 a S_9-B. Pokud zjistí, že těla datových jednotek přenášených v těchto spojích jsou stejná, tak tím si dokáže, že A komunikuje s B. Další možností je, že útočník disponuje rozsáhlými možnostmi (typicky státní zpravodajské služby) a monitoruje provoz ve všech spojích anonymizační sítě. Nejprve podle těl datových jednotek zjistí, že směrovač S_1 přesměroval tyto jednotky ze spoje A-S_1 do spoje S_1-S_2, pak zjistí, že směrovač S_2 sledované datové jednotky přesměroval do spoje S_2-S_5, a tak postupuje dále, až nakonec zjistí, že datové jednotky odeslané stranou A byly směrovačem S_9 předány straně B. Tímto způsobem se tedy opět dá prokázat, že strana A komunikuje se stranou B. K eliminaci uvedených hrozeb se používají tzv. maskovací techniky. Jejich cílem je zajistit anonymitu provozu stran, kdy útočník může monitorováním provozu nanejvýše zjistit, že sledovaná strana A je připojena do anonymizační sítě, avšak při dostatečném počtu aktivních stran nemůže zjistit, s kým vlastně strana A komunikuje.

K vysvětlení maskovacích technik použijeme obr. 2.17. V horní části obrázku vidíme část cesty datových jednotek přenášených od strany A ke straně B. Konkrétně se jedná o první dva spoje, tj. o spoj A-S_1 a S_1-S_2.



Obrázek 2.17: Princip maskovacích technik

Nad přípojným spojem A-S_1 vidíme část časového průběhu přenášených datových jednotek. Jedná se o přenos datové jednotky d_1, za níž po časové mezeře m_1 následuje dvojnásobně delší datová jednotka d_2. Napravo pak vidíme příklad, jaké by mohly být časové poměry datových jednotek d_1 a d_2 ve spoji S_1-S_2, tj. na výstupu směrovače S_1. Z obrázku je zřejmé, že těla datových jednotek jsou stejná (to je reprezentováno stejnou barvou) a mezera m_2 mezi datovými jednotkami je jen mírně odlišná oproti m_1, což je způsobeno náhodnými dobami čekání datových jednotek na přenos ve směrovači S_1. Útočník tak může velmi snadno vysledovat, že ve směrovači S_1 jsou datové jednotky ze spoje A-S_1 přesměrovány do spoje S_1-S_2. Analogicky může postupovat i u ostatních směrovačů, a tak nakonec může zjistit, že strana A komunikuje se stranou B.

Ke ztížení, případně znemožnění analýzy provozu se používají maskovací techniky, k nimž náleží mnohonásobné šifrování, unifikace délek datových jednotek, dávkové zpracování a klamné datové jednotky [14]. Šifrováním těl datových jednotek v každém směrovači sítě se dosáhne toho, že tělo datové jednotky d_i opouštějící směrovač bude zcela jinou pseudonáhodnou posloupností než tělo téže datové jednotky d_i, když do směrovače vstupovala. Útočník tak u tohoto směrovače nebude moci podle obsahu těla datové jednotky přiřadit vstupující datové jednotce d_i odpovídající vystupující datovou jednotku d_i. Nebude tak moci sledovat cestu této datové jednotky sítí. Na dolní části obrázku tuto skutečnost reprezentují různé barvy sobě si odpovídajících jednotek d_i na vstupu a výstupu směrovače S_1. Například datová jednotka d_1 na výstupu tohoto směrovače má jinou barvu (tj. jiný obsah těla) než ta samá datová jednotka d_1 na jeho vstupu.

Další maskovací technikou je unifikace délek datových jednotek. Útočník v důsledku šifrování datových jednotek sice ztratil schopnost ztotožňovat stejné vstupující a vystupující jednotky podle obsahu jejich těla, ale stále je může na vstupu a výstupu směrovače ztotožňovat podle jejich délek. Tím, že komunikace v anonymizační síti bude uskutečňována datovými jednotkami téže délky, útočník tuto možnost ztratí. Na dolní části obrázku uvedenou maskovací techniku ilustrují datové jednotky d_3 a d_4, které vznikly rozdělením datové jednotky d_2 z horní části obrázku.

I v případě unifikace délek datových jednotek však útočník může stále tyto jednotky ztotožňovat podle jejich vzájemných odstupů. Ke ztížení uvedené možnosti lze použít techniku dávkového zpracování, kdy směrovač čeká, až přijme stanovený počet datových jednotek (tzv. dávku) a teprve pak je začne ze své paměti odesílat. Tímto způsobem dojde ke změně časových odstupů mezi vystupujícími datovými jednotkami oproti časovým odstupům mezi vstupujícími datovými jednotkami. Komplementární a bezpečnější variantou, jak útočníkovi znemožnit analýzu toků podle časových odstupů jednotek, je však vkládání falešných datových jednotek. Mírným zpožďováním datových jednotek se upraví mezery mezi datovými jednotkami tak, aby měly délku, která je celistvým násobkem délky datové jednotky. Do těchto mezer se pak vkládají falešné datové jednotky. Přijímací zařízení falešné datové jednotky ignoruje, ale pro pozorovatele vzniká ve spoji situace, že jsou neustále nějaké datové jednotky přenášeny a útočník se pak nemůže v provozu podle žádných mezer orientovat. Například na dolní části obrázku vidíme, že datová jednotka d_2 maskuje v levém, resp. pravém spoji mezeru m_1, resp. m_2.

Různé anonymizační sítě používají různé kombinace výše popsaných maskovacích technik, takže potom také nabízejí i různou úroveň záruky anonymity svých uživatelů. Optimální je kombinace mnohonásobného šifrování, unifikace délek datových jednotek a klamných datových jednotek.

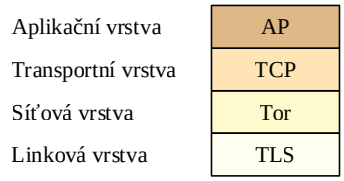
Vysvětlili jsme si obecné základy anonymizačních sítí, a tak se nyní můžeme pustit do samotného popisu sítě Tor. Anonymizační síť Tor („The Onion Routing“) patří zřejmě k nejznámějším a nejvíce používaným anonymizačním sítím. Využívají se v ní následující anonymizační techniky.

- Unifikace délek datových jednotek. Datová jednotka protokolu Tor se nazývá buňka a má jednotnou délku 512 bajtů.
- Substituce adres. Dvojice zdrojová a cílová IP adresa se v každém spoji nahrazuje unikátním identifikátorem v záhlaví buňky.
- Mnohonásobné šifrování datových jednotek. V každém uzlu anonymizační sítě se s každou buňkou provádí šifrování, resp. dešifrování.

Kromě uvedených technik síť Tor umožňuje také klamný provoz, ale ten se v současné době nevyužívá.

Síť Tor aktuálně sestává z cca 6.500 anonymizačních uzlů, využívá ji kolem 2 miliónů uživatelů a slouží ke skrytí zhruba 25 tisíc anonymních severů. Z hlediska bezpečnosti chrání anonymitu uživatelů a serverů vůči lokálním útočníkům (např. zpravodajcům většiny států). Anonymita vůči koalicím útočníků (např. spolupracujícím zpravodajcům více států) garantována není. Koncept sítě Tor vznikl v 90. letech minulého století původně pro zpravodajskou komunitu USA. První verze sítě byla spuštěna v roce 2002 a současná (tj. druhá) funguje od roku 2004.

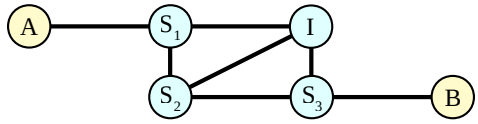
Architektura sítě Tor je velmi podobná architektuře TCP/IP (viz obr. 2.18). Linková vrstva sítě Tor je realizována spoji TLS. Tento protokol zajišťuje vzájemnou autentizaci prvků a ochranu vůči odposlechu i modifikaci přenášených zpráv. Síťová vrstva je řešena protokolem Tor [15]. Tento protokol zajišťuje budování přenosových cest v síti Tor, unifikaci délek přenášených datových jednotek, šifrování a směrování buněk. Transportní službu zajišťuje protokol TCP. V síti Tor lze používat výhradně tento typ transportního protokolu, tj. nelze použít např. protokol UDP. Aplikační vrstvu pak tvoří veškeré aplikace (AP), jejichž komunikační protokol funguje nad protokolem TCP (typicky protokoly HTTP, SMTP, FTP atd.).



Obrázek 2.18: Architektura sítě Tor

Základní prvky sítě Tor vidíme na obr. 2.19. Konkrétně se jedná o následující zařízení.

- Směrovací server S („Onion Router“). Zajišťuje především směrování buněk protokolu Tor v síti. Každý směrovací server S (na našem obrázku jsou tři) má svůj vlastní podepisovací a utajovací kryptosystém.
- Informační server I („Directory Server“). Pro ostatní prvky sítě zajišťuje důvěryhodné informace o síti Tor. Z bezpečnostních důvodů v síti Tor funguje více navzájem nezávislých informačních serverů a každý z nich má svůj podepisovací kryptosystém. Informační servery běží na vybraných směrovacích serverech Tor, takže mohou provádět i směrování.
- Anonymní server B („Hidden Server“). Server, který má být pro své uživatele z hlediska identity majitele i geografického umístění anonymní. Každý anonymní server má svůj podepisovací kryptosystém. Ten se také používá i jako utajovací.
- Klient A („Client“). Počítač těch uživatelů, kteří chtějí zůstat při svém využívání internetu anonymními.

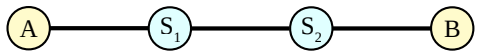


Obrázek 2.19: Základní prvky sítě Tor

V síti Tor se používají utajovací i pečetící kryptosystémy, a to jak symetrického, tak i asymetrického typu. Klíče asymetrických kryptosystémů budeme značit VE_X, SE_X, VP_X a SP_X, kde prefix *V*, resp. *S* vyjadřuje veřejný, resp. soukromý klíč, sufix *E*, resp. *P* reprezentuje klíč utajovacího, resp. autentizačního kryptosystému. Index X pak vyjadřuje majitele klíče, tj. server I, S nebo B. Klíče pro symetrické kryptosystémy se ustavují pomocí Diffie-Hellmanova protokolu.

Autentičnost provozních údajů prvků sítě Tor se zajišťuje podepsanými elektronickými dokumenty. Nejdůležitějšími z nich jsou tzv. deskriptory a konsensus. Deskriptor je výčet údajů o konkrétním směrovači S. Kromě jiného tento výčet obsahuje identifikátor směrovače, jeho veřejný šifrovací klíč VE_S, veřejný ověřovací klíč VP_S a IP adresu směrovače. Deskriptor je podepsán soukromým klíčem SP_S daného směrovače S. Dalším dokumentem je konsensus, což je výčet směrovačů, které většina provozovatelů sítě Tor považuje za důvěryhodné. Konsensus obsahuje i heše ověřovacích klíčů VP_S těchto směrovačů a podepisují jej informační servery I jednotlivých provozovatelů. Veřejné ověřovací klíče VP_I informačních serverů se na anonymních klientech a serverech bezpečně ustaví při instalaci software Tor. Po zmíněné instalaci si uživatelé z informačního serveru stáhnou aktuální konsensus, pomocí klíčů VP_I ověří jeho autentičnost a podle údajů v něm uvedených si pak stahují deskriptory důvěryhodných směrovačů. Autentičnost údajů v deskriptoru pak ověřují pomocí v něm uvedeného klíče VP_S, přičemž heš tohoto klíče musí souhlasit s hešem uvedeným v konsensu. Tímto způsobem se zajišťuje, že případní útočníci nemohou do sítě Tor zapojovat falešné směrovače.

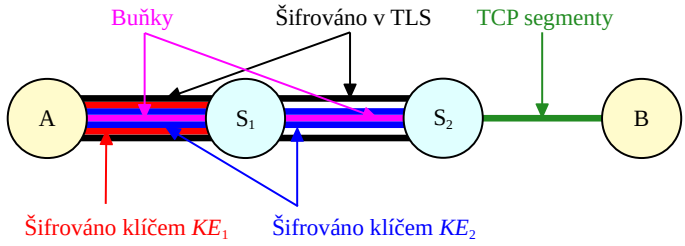
Nyní se pustíme do popisu budování a fungování anonymizujícího spojení od klienta A, přes síť Tor až k veřejnému serveru. Veřejný server zde budeme značit stejně jako anonymní server, tj. označíme jej písmenem B. Dále budeme pro jednoduchost předpokládat, že do tohoto spojení budou zapojeny pouze dva směrovače S_1 a S_2 (viz obr. 2.20).



Obrázek 2.20: Příklad anonymizace klienta A vůči veřejnému serveru B

Anonymizující spoj (dále jen okruh Tor) se buduje postupně. Klient si nejprve vytvoří TLS spoj ke směrovači S_1. Pomocí DH protokolu a šifrovacího kryptosystému směrovače S_1 (viz dále) s ním ustaví šifrovací klíč KE_1 a pečetící klíč KP_1. Buňky pro komunikaci mezi A a S_1 jsou od této chvíle zabezpečovány pomocí uvedených klíčů. Dále si klient A vyžádá na směrovači S_1 prodloužení okruhu ke směrovači S_2. Pokud mezi směrovači S_1 a S_2 ještě spoj TLS neexistuje, tak jej S_1 nejprve vytvoří. Klient pak opět pomocí DH protokolu a šifrovacího kryptosystému směrovače S_2 s tímto směrovačem ustaví šifrovací klíč KE_2 a pečetící klíč KP_2. Buňky pro komunikaci mezi A a S_2 jsou od této chvíle šifrovány a pečetěny těmito klíči. Navíc ve spoji A-S_1 jsou tyto buňky ještě šifrovány klíčem KE_1. Tímto způsobem je v síti Tor uskutečňována anonymizační technika mnohonásobného šifrování. Ve směrovači S_1 jsou buňky pomocí klíče KE_1 dešifrovány a předány směrovači S_2. Ten je pak pomocí klíče KE_2 dešifruje a pomocí klíče KP_2 ověří. Klient si poté od směrovače S_2 vyžádá připojení cílového serveru B, přičemž směrovač S_2 se vůči serveru B tváří jako původní klient.

Komunikace mezi klientem A a serverem B (viz obr. 2. 21) probíhá tak, že spojení mezi A a S_2 se uskutečňuje pomocí buněk protokolu Tor (jedná se o samotný okruh Tor) a spojení mezi S_2 a B se realizuje pomocí segmentů protokolu TCP.

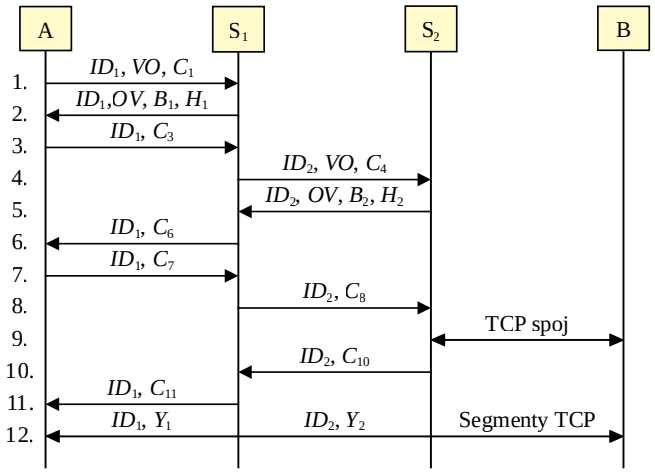


Obrázek 2.21: Příklad mnohonásobného šifrování buněk protokolu Tor

Konverzi dat z buněk na data TCP segmentů a naopak provádí směrovač S_2. Povšimněme si také dvojího šifrování ve spoji A-S_1. Pokud by okruh Tor sestával celkem z n směrovačů, tak by v něm

klient měl sjednány šifrovací klíče KE_1 až KE_n. Ve spoji A-S_1 by buňky byly zašifrovány nejprve klíčem KE_n, takto vzniklý kryptogram pak klíčem KE_{n-1} a tak dále. Posledním z mnohonásobného šifrování buňky by bylo šifrování klíčem KE_1. Zkráceně to označíme, že buňka je ve spoji A-S_1 mnohonásobně šifrována posloupností klíčů (KE_n, KE_{n-1}, ..., KE_2, KE_1). Ve směrovači S_1 je buňka dešifrována klíčem KE_1 a předána do spoje S_1-S_2. V tomto spoji tedy bude buňka zašifrována posloupností klíčů (KE_n, KE_{n-1}, ..., KE_2). V každém dalším směrovači bude buňka dešifrována příslušným klíčem, takže nakonec ve směrovači S_n bude již buňka v nezašifrované podobě. V našem příkladu n = 2, a tak v prvním spoji je dvojí šifrování posloupností klíčů (KE_2, KE_1) a ve druhém spoji jedno šifrování klíčem KE_1. Popsané postupné dešifrování buněk po jednotlivých klíčích připomínalo tvůrcům sítě Tor loupání cibule po jednotlivých vrstvách, a tak síť do svého názvu dostala slovo cibule („onion“).

Nyní si vytvoření okruhu Tor (zkráceně OT) popíšeme podrobněji (viz obr. 2.22). Předpokládejme, že klient A si pro vytvoření okruhu k veřejnému serveru B vybral směrovače S_1 a S_2 a z jejich deskriptorů zná jejich veřejné šifrovací klíče VE_1 a VE_2.



Obrázek 2. 22: Vytvoření a fungování okruhu Tor

1. V prvním kroku klient A nejprve vybuduje ke směrovači S_1 spoj TLS a v něm odešle buňku Tor, v jejímž záhlaví bude identifikátor ID_1. V těle buňky je příkaz VO („Vytvoř okruh“) a kryptogram C_1 = ENC(A_1, VE_1), kde A_1 = DHF(V, a_1). Hodnota V je veřejně známý vzor DH protokolu a a_1 je první DH klíč klienta. Směrovač S_1 nyní ví, že v daném TLS spoji bude budován okruh a že buňky tohoto okruhu se od případných buněk jiných okruhů budou odlišovat pomocí ID_1. Svým soukromým klíčem SE_1 nyní dešifruje kryptogram C_1 a získá tak DH obraz A_1 = DEC(C_1, SE_1). Směrovač S_1 vygeneruje svůj DH klíč b_1 a odvodí tajné semeno S_1 = DHF(A_1, b_1).

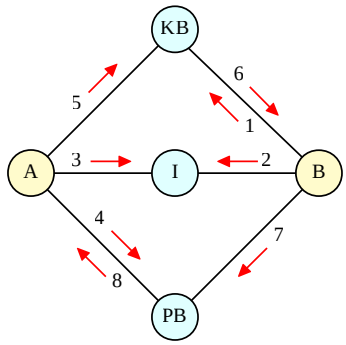
2. Směrovač S_1 odpoví klientovi A buňkou, v jejímž záhlaví bude identifikátor ID_1 a v těle se bude nacházet hlášení *OV* („Okruh vytvořen“) spolu s DH obrazem $B_1 = \mathrm{DHF}(V, b_1)$ a kontrolním hešem $H_1 = \mathrm{HSF}(S_1)$. Nyní si i klient může odvodit semeno $S_1 = \mathrm{DHF}(B_1, a_1)$ a jeho správnost ověřit tím, že heš $\mathrm{HSF}(S_1)$ je roven přijaté hodnotě H_1. Tím je zároveň směrovač S_1 autentizován. Obě strany ze semene odvodí pomocí odvozovací funkce ODF šifrovací klíč KE_1 a pečetící klíč KP_1. Platí, že $(KE_1, KP_1) = \mathrm{ODF}(S_1, T)$, kde T je stanovený kontext. Klient A bude klíč KE_1 používat k šifrování všech buněk odesílaných do spoje A-S_1 a k dešifrování všech buněk přijatých ze spoje A-S_1. V případě směrovače S_1 tomu bude naopak.
   Klíč KP_1 budou obě strany používat k pečetění, resp. ověřování jen těch buněk, které slouží k jejich vzájemné komunikaci. V buňce totiž může být vždy jen jediná pečeť P, která se zpravidla využívá koncovými stranami okruhu (tj. klientem a v našem příkladu směrovačem S_2). Tranzitní směrovače (v našem případě S_1) poznávají komunikaci od klienta, která je pro ně určena tak, že pro každou dešifrovanou buňku ze směru od klienta vypočítají svoji hodnotu pečeti P'. Pokud $P' = P$, tak je buňka součástí interní komunikace daného směrovače s klientem A, tj. adresátem buňky je tranzitní směrovač. V opačném případě se dešifrovaná buňka předá dalšímu směrovači. Kontrolním výpočtem hodnoty P' identifikuje případné buňky s komunikací od tranzitních směrovačů i klient. Pro zjednodušení dalšího výkladu budování okruhu Tor však toto testování hodnot pečetí již popisovat nebudeme.
3. Ve třetím kroku protokolu klient odešle buňku s identifikátorem ID_1, v jejímž těle se nachází kryptogram $C_3 = \mathrm{ENC}(Z_3, KE_1)$. Zpráva Z_3 obsahuje příkaz *PO* („Prodluž okruh“) k S_2 a kryptogram $C_4 = \mathrm{ENC}(A_2, VE_2)$, kde $A_2 = \mathrm{DHF}(V, a_2)$, přičemž a_2 je druhý DH klíč klienta. Směrovač S_1 kryptogram C_3 dešifruje, čímž získá zprávu Z_3. Dozví se z ní, že má prodloužit okruh ke směrovači S_2 a tomu má předat kryptogram C_4.
4. Pokud mezi S_1 a S_2 doposud neexistuje TLS spoj, tak jej S_1 nejprve vybuduje. Směrovač S_1 vygeneruje pro tento spoj unikátní identifikátor ID_2, jímž bude odlišovat buňky budovaného okruhu OT od buněk případných dalších okruhů. Do své směrovací tabulky si pak zapíše, že těla buněk s identifikátorem ID_1 ze spoje A-S_1 má po dešifrování odesílat do spoje S_1-S_2 v buňkách s identifikátorem ID_2 a naopak. Následně směrovač S_1 odešle do TLS spoje k S_2 buňku s identifikátorem ID_2, příkazem *VO* („Vytvoř okruh“) a kryptogramem C_4.
5. Směrovač S_2 si zaeviduje, že v tomto TLS spoji bude budován okruh a že buňky tohoto okruhu bude od ostatních odlišovat pomocí ID_2. Svým soukromým klíčem SE_2 dešifruje kryptogram C_4, čímž získá DH obraz $A_2 = \mathrm{ENC}(C_4, SE_2)$. Následně vygeneruje svůj DH klíč b_2 a odvodí semeno $S_2 = \mathrm{DHF}(A_2, b_2)$. Směrovač S_2 poté odpoví směrovači S_1 buňkou s ID_2, v jejímž těle bude hlášení *OV* („Okruh vytvořen“) spolu s DH obrazem $B_2 = \mathrm{DHF}(V, b_2)$ a kontrolním hešem $H_2 = \mathrm{HSF}(S_2)$.
6. Směrovač S_1 poté zašle klientovi buňku s ID_1 a kryptogramem $C_6 = \mathrm{ENC}(Z_6, KE_1)$, kde zpráva Z_6 obsahuje hlášení *OP* („Okruh prodloužen“), DH obraz B_2 a kontrolní heš H_2. Klient kryptogram C_6 dešifruje, čímž obdrží hlášení o prodloužení okruhu k S_2 a hodnoty B_2 a H_2. Nyní si i klient může odvodit semeno $S_2 = \mathrm{DHF}(B_2, a_2)$, přičemž si jeho správnost ověří kontrolním hešem H_2. Obě strany (tj. A a S_2) ze semene odvodí pomocí odvozovací funkce ODF šifrovací klíč KE_2 a pečetící klíč KP_2.

7. V následujícím kroku protokolu zašle klient směrovači S_1 buňku s ID_1 a kryptogramem C_7 = ENC(Z_7, KE_2 až KE_1), kde parametry KE_2 až KE_1 vyjadřují, že zpráva Z_7 je mnohonásobně zašifrována v pořadí klíčů od KE_2 po KE_1. Zpráva Z_7 přitom obsahuje příkaz *VS* („Vybuduj spoj“) k serveru B.
8. Směrovač S_1 kryptogram C_7 dešifruje, čímž získá kryptogram C_8 = DEC(C_7, KE_1) = ENC(Z_7, KE_2). Ze své směrovací tabulky podle ID_1 zjistí, že získaný kryptogram C_8 má v buňce s ID_2 předat směrovači S_2. Kryptogram tedy odešle.
9. Směrovač S_2 přijatý kryptogram dešifruje a získá tak zprávu Z_7 = DEC(C_8, KE_2), čímž zjistí, že má vybudovat TCP spoj k veřejnému serveru B. Požadované spojení vybuduje.
10. Směrovač S_2 poté vytvoří zprávu Z_{10}, která obsahuje hlášení *SV* („Spoj vybudován“). Uvedenou zprávu zašifruje svým klíčem KE_2, čímž vznikne kryptogram C_{10} = ENC(Z_{10}, KE_2). Tento kryptogram odešle v buňce s ID_2 směrovači S_1.
11. Směrovač S_1 přijatý kryptogram zašifruje klíčem KE_1, čímž vznikne kryptogram C_{11} = ENC (C_{10}, KE_1) = ENC(Z_{10}, KE_2 až KE_1). Podle směrovací tabulky tento kryptogram odešle v buňce s ID_1 klientovi A.
12. Klient přijatý kryptogram dešifruje a získá Z_{10} = DEC(C_{11}, KE_1 až KE_2). Z této zprávy se dozví, že spojení k serveru B je vybudováno a lze s ním zahájit komunikaci. Komunikace mezi klientem A a serverem B probíhá oboustranně tak, že přenášená data jsou členěna na bloky *D*, k nimž se vypočítává pečeť P = PCT(D, KP_2). Data D a pečeť P se zřetězí a tyto řetězce jsou ve spoji A-S_1 přenášeny v buňkách s ID_1 v podobě kryptogramů Y_1 = ENC($D \parallel P$, KE_2 až KE_1) a ve spoji S_1-S_2 jsou přenášeny v buňkách s ID_2 v podobě kryptogramů Y_2 = ENC($D \parallel P$, KE_2). Ve směrovači S_2 se po dešifrování zkontroluje správnost pečeti a data D z jednotlivých buněk se uskupují do podoby datových segmentů, které pak jsou protokolem TCP přeneseny k serveru B. V opačném směru přenosu je postup analogický.

Z předchozího popisu sítě Tor vidíme, že tato síť je doslova prošpikována kryptografickými technikami. Digitální podpisy zabraňují útočníkům oklamat uživatele falešnou sítí, asymetrické kryptosystémy znemožňují směrovačům Tor vystupovat jako jiné směrovače, DH protokol je použit k ustavení klíčů pro symetrické kryptosystémy, symetrické šifry zajišťují jak utajení přenášených dat, tak i modifikaci těl buněk a pečetící kryptosystémy slouží k ověřování autentičnosti přenášených dat.

Z dosavadního popisu je rovněž zřejmé, že síť Tor využívá techniku přenosu dat datovými jednotkami unifikované délky, využívá i techniku změn datové podoby těl buněk pomocí jejich mnohonásobného šifrování a rovněž tak využívá techniku substituce adres, kdy dvojice adres klient A a server B je v každém spoji nahrazena unikátním identifikátorem *ID* v záhlaví buňky. Rovněž tak jsme viděli, že jednotlivé směrovače vždy z celého spojení mezi klientem A a serverem B znají jen předchozí a následující prvek. V našem příkladu tak směrovač S_1 ví, že klient A požaduje spojení se směrovačem S_2. Nemůže však z následující kryptograficky zabezpečené komunikace zjistit, že klient A si nakonec vyžádal spojení se serverem B. Podobně směrovač S_2 jen ví, že kdosi komunikuje přes směrovač S_1 se serverem B.

Poslední věcí, která nám pro popis sítě Tor zůstává k vysvětlení, je fungování anonymního serveru, tj. serveru, který poskytuje anonymitu svému provozovateli a zároveň nabízí i skrytí geografické polohy tohoto serveru. Principem fungování (viz obr. 2.23) je, že klient A oznámí anonymnímu serveru B prostřednictvím kontaktního bodu KB (serverem vybraný směrovač), že se k němu chce připojit v propojovacím bodě PB (klientem vybraný směrovač). K provozu anonymního serveru je zapotřebí vytvořit šifrovací asymetrický kryptosystém se soukromým klíčem SE_B a veřejným klíčem VE_B. Tento kryptosystém je kromě šifrování zpráv používán také k podepisování (viz dále). Veřejný klíč a další údaje o serveru B jsou dostupné přes informační servery I. V souvislosti s obrázkem ještě upozorňujeme, že hrany grafu nejsou jednotlivé spoje TLS, ale celé okruhy Tor.



Obrázek 2.23: Anonymizace serveru

Anonymizace serveru probíhá následovně.

1. Server B si vybere několik směrovačů, které budou plnit roli jeho kontaktního bodu KB. Ke každému z nich vybuduje okruh, ve kterém mu zašle žádost o plnění zmiňované role. Žádost obsahuje veřejný klíč VE_B serveru a je podepsána jeho soukromým klíčem SE_B. Oslovený směrovač tento podpis ověří a svůj souhlas s rolí KB pro server B potvrdí. Poznamenáváme, že použitý kryptosystém je sice šifrovací, ale tyto lze využít také k podpisování. Princip je takový, že podpisem zprávy Z je heš této zprávy zašifrovaný soukromým klíčem SE_B. Formálně zapsáno platí, že podpis $P = \mathrm{ENC}(H, SE_B)$, kde $H = \mathrm{HSF}(Z)$. Kdokoliv pak může z podepsané zprávy (Z, P) zjistit jak vlastní hodnotu heše $H' = \mathrm{HSF}(Z)$, tak i zjistit hodnotu heše $H = \mathrm{DEC}(P, VE_B)$, k níž dospěl autor podpisu. Pokud platí, že $H' = H$, tak je zpráva autentická.
2. Když anonymní server B získal souhlas dostatečného počtu kontaktních bodů, tak vytvoří svůj deskriptor. Tento deskriptor obsahuje veřejný klíč VE_B serveru a seznam kontaktních bodů. Podepsán je soukromým klíčem SE_B. Server B poté postupně vybuduje okruhy ke všem informačním serverům I (na našem obrázku máme jediného zástupce) a svůj deskriptor jim předá. Z heše veřejného klíče VE_B serveru se odvodí řetězec o délce 16 alfanumerických

znaků, který spolu s příponou „onion“ tvoří unikátní jméno JM_B daného serveru (např. 234567ABCDEFGHIJ.onion). Servery I posléze jméno JM_B (a tedy zároveň heš VE_B) zahrnou do konsensu a deskriptor serveru zpřístupní uživatelům.

3. Pokud chce uživatel využívat služby serveru B, tak do adresního řádku svého prohlížeče Tor napíše jméno JM_B serveru. Počítač A tohoto uživatele následně vybuduje okruh k některému z informačních serverů a z jeho adresáře si stáhne deskriptor serveru B. Autentičnost deskriptoru ověří pomocí v něm uvedeného klíče VE_B a zkontroluje, zda heš tohoto klíče dává po překódování jméno JM_B uvedené v konsensu.
4. Klient si poté zvolí směrovač S, v němž dojde k propojení se serverem B (tzv. propojovací bod – PB). Klient vybuduje k PB okruh, požádá jej o zprostředkování komunikace a předá mu tajné autentizační heslo *AH*.
5. Následně klient z deskriptoru serveru B vybere jeden z uváděných kontaktních bodů KB, vybuduje k němu okruh a v něm mu předá kryptogram $C_5 = \mathrm{ENC}(Z_5, VE_B)$. Zpráva $Z_5 = (\mathrm{PB}, AH, A)$, kde PB je identifikátor propojovacího bodu PB, *AH* je autentizační heslo a hodnota *A* je DH obraz klienta. Platí tedy, že $A = \mathrm{DHF}(V, a)$, kde *V* je veřejně známý DH vzor a *a* je tajný DH klíč klienta.
6. Kontaktní bod KB kryptogram C_5 předá v předem připraveném okruhu (viz 1. krok) serveru B. Server B svým soukromým klíčem SE_B kryptogram dešifruje, a tak získá trojici $\mathrm{DEC}(C_5, SE_B) = (\mathrm{PB}, AH, A)$.
7. Server B nyní zvolí svůj DH klíč *b* a vypočítá semeno $S = \mathrm{DHF}(A, b)$. Pro protistranu také vypočítá DH obraz $B = \mathrm{DHF}(V, b)$. Následně ke klientem udanému propojovacímu bodu PB vybuduje okruh a v něm mu předá dvojici hodnot (*AH*, *B*).
8. Propojovací bod zkontroluje správnost hesla *AH* (zná jej ze 4. kroku) a klientovi předá v již vybudovaném okruhu DH obraz *B*. Klient tak může sestrojit semeno $S = \mathrm{DHF}(B, a)$. Ze společného semena pak obě strany odvodí klíč pro šifrování *KE* a klíč pro pečetění *KP*, přičemž $(KE, KP) = \mathrm{ODF}(S, T)$, kde *T* je stanovený kontext.

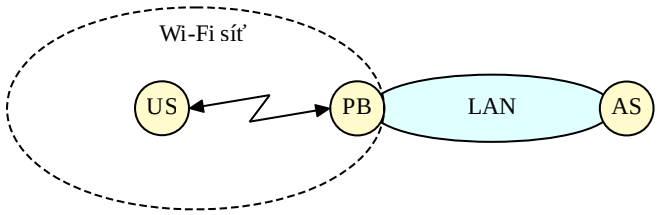
Nyní mohou klient A a server B vést přes propojovací bod PB vzájemnou komunikaci. Například ve směru od klienta A se data z buňky nejprve pomocí klíče *KP* opatří pečetí, dále zašifrují klíčem *KE* a poté se ještě zašifrují postupně všemi klíči (KE_n, KE_{n-1}, ..., KE_2, KE_1) všech *n* směrovačů v okruhu z A do PB. Cestou tímto okruhem jsou buňky postupně dešifrovány, až v uzlu PB jsou zašifrovány jen klíčem *KE*. Buňky se poté v PB odesílají do okruhu z PB do B. Na své cestě do B jsou postupně šifrovány klíči (KE_m, KE_{m-1}, ..., KE_2, KE_1) všemi *m* směrovači v tomto okruhu. V serveru B jsou buňky postupně všemi těmito klíči dešifrovány a nakonec jsou ještě dešifrovány i klíčem *KE*. Poté proběhne ověření pečeti a když je vše v pořádku, tak jsou data z buněk předány k dalšímu zpracování. V opačném směru je postup analogický.

2.7 Kryptografie v linkové vrstvě

Tato podkapitola je věnována kryptograficky zabezpečeným přenosům v linkové vrstvě. Vysvětlíme si zde komplex WPA a protokol MACsec.

2.7.1 Komplex WPA

Komplex WPA („Wi-Fi Protected Access“) je soubor kryptografických protokolů pro zabezpečení Wi-Fi spojů, což jsou rádiové spoje definované standardem IEEE 802.11. V praxi se zcela nejčastěji potkáváme s konfigurací Wi-Fi spoje (viz obr. 2.24), kdy uživatelská stanice US (dále také strana A) se pomocí rádiového spoje připojuje k tzv. přístupovému bodu PB (dále také strana B). Přístupový bod uživatelské stanici zprostředkovává připojení k dalším uživatelským stanicím dané Wi-Fi sítě a rovněž jí zprostředkovává přístup do jiných sítí (obvykle přes lokální síť LAN). Na obrázku ještě vidíme autentizační server AS (dále také strana C), který je důležitý pro provoz Wi-Fi sítí u velkých organizací (viz dále).



Obrázek 2.24 Struktura a místo Wi-Fi sítě

Komplex WPA nahradil původní kryptografický protokol pro Wi-Fi spoje, který nesl označení WEP („Wired Equivalent Privacy“). V současné době jsou standardizovány tři verze komplexu WPA, které jsou označovány zkratkami WPA, WPA2 a WPA3, přičemž se doporučuje používat poslední dva uvedené. My si zde popíšeme nejnovější verzi, tj. komplex WPA3 [16].

Zabezpečení v rámci komplexu WPA3 lze rozčlenit do tří fází. V první (nazveme ji ověřovací) fázi dojde ke vzájemné autentizaci stran a ke sjednání společného semena, které se ve standardu označuje zkratkou *PMK* („Pairwise Master Key“). Autentizace přitom může být buď lokální (tj. v rámci dané sítě), nebo centralizovaná (společná pro více sítí). Poté následuje fáze, kterou budeme nazývat ustavovací fáze („4-way Handshake“). V jejím průběhu dojde k ustavení bloku klíčů, který se označuje zkratkou *PTK* („Pairwise Transient Key“). Třetí fázi pojmenujeme provozní – je to fáze samotného šifrování a pečetění rámců. K tomu se využívá integrovaná symetrická kryptografie.

Jak již bylo uvedeno, ověřovací fáze má dvě varianty autentizace. V případě lokální varianty se uživatelská stanice US autentizuje vůči přístupovému bodu PB sítě a v případě centralizované varianty se uživatelská stanice autentizuje vůči centrálnímu autentizačnímu serveru AS. První varianta je určena pro malé sítě, tj. pro sítě fyzických osob, nebo malých firem. V tomto případě se všechny uživatelské stanice a přístupový bod navzájem autentizují na základě znalosti sdíleného hesla. Výhodou uvedené varianty je snadné nastavení a skutečnost, že nevyžaduje autentizační server. Slabinou této varianty je sdílené heslo. Pokud dojde k jeho vyzrazení, tak jsou ohroženi všichni uživatelé sítě.

Ve druhé variantě se uživatelská stanice autentizuje vůči centrálnímu autentizačnímu serveru, přičemž tato autentizace je individuální, tj. například v případě hesel se každý uživatel autentizuje svým vlastním heslem. Uvedený způsob autentizace je výhodný pro velké firmy a instituce, v nichž mohou být provozovány až desítky různých Wi-Fi sítí a stovky uživatelských stanic. Předností popsané varianty je vyšší úroveň bezpečnosti (prozrazením hesla jednoho z uživatelů nejsou ohroženi ostatní uživatelé Wi-Fi sítě) a provozní jednoduchost (v přístupových bodech není nutné instalovat, chránit a aktualizovat seznamy hesel uživatelů). Nevýhodou varianty je nutnost provozovat autentizační server. V této souvislosti je zapotřebí poznamenat, že i v případě varianty s autentizačním serverem musí ještě proběhnout autentizace mezi uživatelskou stanicí a přístupovým bodem. Ta je součástí až ustavovací fáze, kterou probereme později.

Nejprve si vysvětlíme ověřovací protokol s lokální variantou autentizace. V komplexu WPA3 se tato varianta označuje jako protokol SAE („Simultaneous Authentication of Equals"), přičemž autentizace je zde založena na znalosti sdíleného síťového hesla. Protokol SAE kromě vzájemné autentizace uživatelské stanice sítě (strana A) a přístupového bodu (strana B) rovněž zajišťuje sjednání tajného semena *PMK*.

Protokol SAE je prakticky DH protokol rozšířený o autentizaci stran, takže se zde opět potkáme s DH klíči, vzory i obrazy (viz podkapitola DH protokol) a i zde budeme pro zjednodušení popisu vynechávat zápis operace modulo. Strana A i strana B znají společné síťové heslo *PSW*, které si před zahájením samotného provozu převedou na DH vzor $X = \mathrm{F}(PSW)$, kde F je standardem WPA3 definovaný algoritmus konverze textového hesla *PSW* na DH vzor X (tj. na číslo, nebo na bod eliptické křivky). Následující výměnu zpráv ověřovacího protokolu SAE pak můžeme sledovat na obr. 2.25.

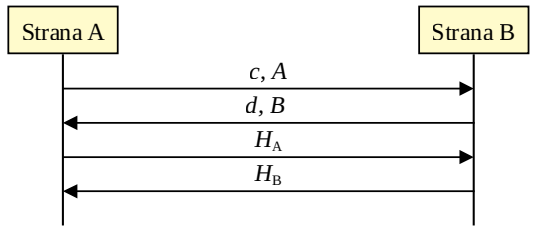
Po navázání spojení strana A nejprve vygeneruje náhodný tajný klíč a a tzv. maskovací klíč u. DH klíč a následně zašifruje přičtením maskovacího klíče u do podoby kryptogramu $c = a+u$. Dále strana A vypočítá DH obraz $A = X^{-u}$ a dvojici (c, A) odešle straně B. Strana B si po přijetí uvedené dvojice rovněž vygeneruje náhodný tajný klíč b a svůj maskovací klíč v. DH klíč b podobně jako strana A zašifruje přičtením maskovacího klíče v do podoby kryptogramu $d = b+v$. Dále strana B vypočítá svůj DH obraz $B = X^{-v}$ a dvojici (d, B) odešle straně A.

Obě strany nyní mohou určit hodnotu tajného semene S. Strana A přijala dvojici (d, B) a z tajného DH vzoru hesla X a ze svého klíče a vypočítá

$$(X^{d} \cdot B)^{a} = (X^{b+v} \cdot X^{-v})^{a} = (X^{b+v-v})^{a} = (X^{b})^{a} = X^{b \cdot a} = S$$

a strana B analogicky dospěje k téže hodnotě

$$(X^{c} \cdot A)^{b} = (X^{a+u} \cdot X^{-u})^{b} = (X^{a+u-u})^{b} = (X^{a})^{b} = X^{a \cdot b} = S.$$



Obrázek 2.25: Výměna zpráv u ověřovacího protokolu SAE

Ze získaného DH semena *S* obě strany pomocí určené odvozovací funkce ODF odvodí další dvě semena, která jsou ve standardu označena zkratkami *KCK* a *PMK*. Platí, že (*KCK* ‖ *PMK*) = ODF(*S*, *T*), kde kontext *T* stanoveným způsobem závisí na součtu (*c*+*d*), což formálně vyjádříme funkcí *T* = f(*c*+*d*). Semeno *PMK* je výstupem protokolu SAE a semeno *KCK* je použito k vytvoření níže uvedených zpráv.

K potvrzení toho, že obě strany dospěly k téže hodnotě semene *S*, si tyto strany vymění kontrolní heše H_A a H_B. Strana A spočítá heš H_A = HSF(*KCK* ‖ *c* ‖ *A* ‖ *d* ‖ *B*) a odešle jej straně B. Ta provede kontrolní výpočet, a pokud se dopracuje rovněž k hodnotě H_A, tak získá záruku, že protistrana zná tajnou hodnotu *KCK* a tedy, že zná i tajné semeno *S*. Tímto způsobem se strana A autentizuje, protože ten, kdo vypočítá správnou hodnotu *S*, musí také znát i tajný DH vzor *X* daný síťovým heslem *PSW*.

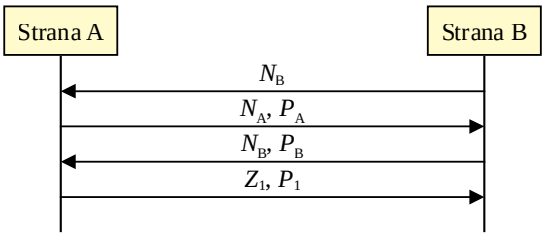
Strana B nyní spočítá heš H_B = HSF(*KCK* ‖ *d* ‖ *B* ‖ *c* ‖ *A*) a odešle jej straně A. Ta rovněž provede svůj kontrolní výpočet, a pokud se dopracuje k téže hodnotě H_B, tak získá záruku, že protistrana B zná tajnou hodnotu semene *S*, a tedy i ona zná tajný DH vzor *X*. Tím se strana B autentizovala vůči straně A. V tomto okamžiku protokol SAE končí. Vidíme, že obě strany se v jeho průběhu navzájem autentizovaly a dokázaly protistraně, že náleží do téže Wi-Fi sítě. Kromě toho odvodily i tajné semeno *PMK*.

Po skončení protokolu SAE následuje fáze ustavení klíče. Před jejím popisem si však ještě musíme vysvětlit variantu ověřovací fáze s centralizovanou autentizací. I v tomto případě se uživatelská stanice (tj. strana A) připojuje k připojovacímu bodu (strana B). Připojovací bod je však v této variantě ověřovací fáze pasivní a jen zprostředkovává výměnu zpráv mezi stranou A a autentizačním serverem (strana C). V rámci této výměny zpráv dojde ke vzájemné autentizaci stran A a C a ke sjednání hodnoty semena *PMK*.

Centralizovaná autentizace je založena na protokolu EAP („Extensible Authentication Protocol“), který si podrobněji vysvětlíme v následující kapitole. Zde si pro ilustraci pouze stručně popíšeme princip varianty, která je označována zkratkou PEAP („Protected Extensible Authentication Protocol“). V uvedeném případě si strany A a C vyměňují zprávy protokolu EAP, v jejichž těle se

nacházejí zprávy nám již známého protokolu TLS. Tímto způsobem fakticky dojde k vytvoření kryptograficky zabezpečeného TLS spojení mezi A a C. V rámci zahajovací fáze TLS protokolu se C (tj. autentizační server) autentizuje certifikátem svého veřejného klíče. Strana A se pak v rámci navázaného šifrovaného spojení TLS autentizuje vybranou EAP metodou (např. pomocí EAP-MD5). Z popisu protokolu TLS víme, že během jeho zahajovací fáze dojde k odvození společného semene *S*. Z tohoto semena se v protokolu PEAP odvodí tajná hodnota, která se v protokolu EAP označuje jako *MSK* („Master Session Key"). Hodnota *MSK* je pro protokol WPA3 hodnotou *PMK*. Tuto hodnotu strana C šifrovaně předá přístupovému bodu, tj. straně B. Strana B se tak dozví, že autentizace strany A proběhla v pořádku a ví, že nyní spolu se stranou A sdílí tajné semeno *PMK*. Na základě této hodnoty se strany A i B budou v ustavovací fázi navzájem autentizovat a odvodí si potřebné klíče.

Nyní se v komplexu WPA3 vraťme k fázi ustavení klíče. V této fázi se používá protokol, který označíme zkratkou 4WH („4-way Handshake"). Již jsme si uvedli, že výsledkem obou výše popsaných variant zahajovacího protokolu (jak lokální SAE, tak i centralizované EAP) je, že strana A i strana B sdílejí tajné semeno *PMK*. Ve fázi ustavení klíčů se z hodnoty *PMK* určí potřebné klíče a v případě centralizované autentizace v této fázi dojde i ke vzájemné autentizaci uživatelské stanice A a připojovacího bodu B. Výměnu zpráv v rámci ustavovacího protokolu 4WH ilustruje obr. 2.26.



Obrázek 2.26: Výměna zpráv ustavovacího protokolu 4WH

Na začátku ustavovacího protokolu 4WH strana B vygeneruje a protistraně odešle náhodné číslo N_B. Po jeho přijetí strana A vygeneruje své náhodné číslo N_A a vypočítá blok PTK = ODF(PMK, T), kde ODF je stanovená odvozovací funkce, PMK je tajné semeno a T = f(N_A, N_B) je kontext závislý na náhodných číslech N_A a N_B. Blok PTK sestává z klíčů KP, KE a KH, tj. $PTK = KP \parallel KE \parallel KH$. Klíč KP, resp. KE je pečetící, resp. šifrovací klíč pro ustavovací fázi a klíč KH je hlavní klíč, který je určený pro šifrování a pečetění ve fázi provozu.

Strana A následně protistraně B odešle své náhodné číslo N_A spolu s jeho pečetí P_A = PCT(N_A, KP). Z čísla N_A nyní může i strana B odvodit blok PTK, a tak získat klíče KP, KE a KH. Pomocí pečetícího klíče KP následně ověří pečeť P_A. Pokud VER(N_A, P_A, KP) = 1, tak tím strana B získala

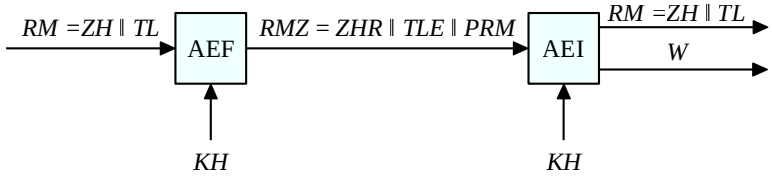
záruku, že protistrana vypočítala správnou hodnotu *KP*. V případě centralizované autentizace navíc správná hodnota *KP* implikuje, že strana A zná hodnotu tajného semena *PMK*. Strana B (tj. připojovací bod) si tímto způsobem ověřila, že strana A je skutečně ta uživatelská stanice, které během zahajovací fáze zprostředkovávala autentizaci vůči autentizačnímu serveru C. Tím je dokončena autentizace strany A vůči straně B.

Strana B nyní protistraně znovu odešle své číslo N_B, tentokráte však i s jeho pečetí $P_B = \mathrm{PCT}(N_B, KP)$. Strana A pomocí klíče *KP* ověří P_B. Pokud $\mathrm{VER}(N_B, P_B, KP) = 1$, tak strana A získala záruku, že protistrana B vypočítala správnou hodnotu klíče *KP*. V případě centralizované autentizace se tím zároveň strana B autentizuje vůči straně A, neboť znalost *KP* předpokládá znalost tajné hodnoty *PMK*. Uživatelská stanice tak má nyní záruku, že se připojuje do sítě provozovatele, který zároveň provozuje i autentizační server C.

Přenosem dvojice N_B a P_B strana B zároveň protistraně signalizuje, že je připravena na přechod do provozní fáze. Signál o své připravenosti k provozu vytvoří nyní i strana A tím, že vytvoří a odešle zprávu Z_1 o ukončení ustavovací fáze spolu s její pečetí $P_1 = \mathrm{PCT}(Z_1, KP)$. Strana A poté přejde do provozní fáze. Strana B standardním způsobem ověří správnost pečeti P_1, a pokud je zpráva Z_1 autentická, tak do provozní fáze přejde rovněž.

V provozní fázi jsou Wi-Fi rámce šifrovány a pečetěny hlavním klíčem *KH*. K šifrování a pečetění rámců povoluje standard WPA3 tři kryptosystémy, které jsou označovány zkratkami TKIP, CCMP a GCMP. Fakticky se ve všech třech případech jedná o symetrické integrované kryptosystémy, které navíc zajišťují i autentizaci služebních údajů ze záhlaví rámce.

K obecnému popisu integrovaného kryptosystému pro WPA3 využijeme autentizační a pečetící funkci AEF a k ní inverzní funkci AEI (viz obr. 2.27). Formálně pak můžeme zabezpečení rámců vyjádřit následovně. Má-li se přenést standardní Wi-Fi rámec $RM = ZH \parallel TL$, kde *ZH* je záhlaví rámce a *TL* je tělo rámce, tak se vytvoří zabezpečený WPA rámec $RMZ = ZHR \parallel TLE \parallel PRM$, kde *ZHR* je rozšířené záhlaví (obsahuje navíc služební kryptografické údaje), *TLE* je zašifrované tělo a *PRM* je pečeť celého rámce. Odesílající strana zabezpečený rámec vypočítá tak, že $RMZ = \mathrm{AEF}(RM, KH)$ a tento rámec pak zašle protistraně, která provede inverzní operace, tj. vypočítá dvojici $(RM, W) = \mathrm{AEI}(RMZ, KH)$. Pokud pro autentizační indikátor platí, že $W = 1$, tak je výstupní rámec *RM* předán ke standardnímu zpracování.

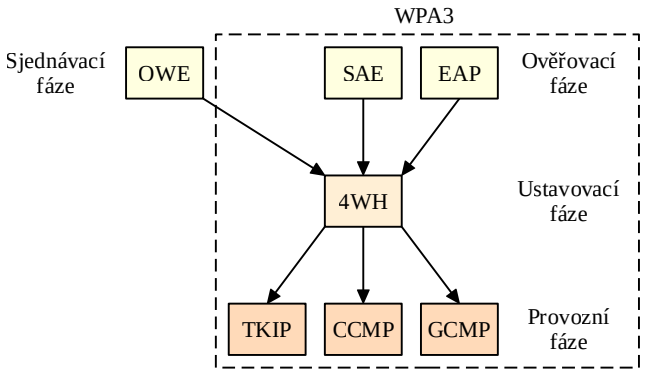


Obrázek 2.27: Schéma integrovaného symetrického kryptosystému pro WPA3

S kryptografickým zabezpečením Wi-Fi sítí souvisí také protokol, který je označován zkratkou OWE („Opportunistic Wireless Encryption“) [17]. Tento prokol je alternativou ověřovacího protokolu SAE, resp. EAP, avšak jeho účelem není zajistit autentizaci stran. Jeho účelem je zajistit pouze sjednání tajného semena *PMK*, ze kterého se pak v rámci ustavovacího protokolu 4WH odvodí hlavní klíč *KH*. Ten je pak použit pro šifrování a pečetění rámců. Protokol OWE je určen pro Wi-Fi sítě kaváren, obchodních domů apod., jejichž návštěvníci mohou prostřednictvím Wi-Fi sítě získat přístup k internetu a dalším službám (tzv. veřejné Wi-Fi sítě). Provozovatelé uvedených sítí nepotřebují uživatele své sítě autentizovat, na druhou stranu však kryptograficky zabezpečený provoz ochrání uživatele sítě před útoky třetích stran.

Protokol OWE je prakticky DH protokol mezi uživatelskou stanicí (strana A) a přístupovým bodem (strana B). Po navázání spojení strana A vygeneruje svůj DH klíč a a vypočítá svůj DH obraz $A = \mathrm{DHF}(V, a) = V^a$, kde vzor V je veřejně známá hodnota. DH obraz A poté odešle straně B. Strana B vygeneruje svůj DH klíč b a vypočítá DH obraz $B = \mathrm{DHF}(V, b) = V^b$, který odešle straně A. Obě strany nyní mohou odvodit společné semeno *PMK*. Strana A vypočítá $PMK = \mathrm{DHF}(B, a) = B^a = (V^b)^a = V^{b \cdot a}$ a strana B se analogicky dopočítá k téže hodnotě $PMK = \mathrm{DHF}(A, b) = A^b = (V^a)^b = V^{a \cdot b}$. V tomto okamžiku protokol OWE končí a obě strany přejdou do ustavovací fáze (protokol 4WH), během níž ustaví hlavní klíč *KH* a následně pak přejdou do provozní fáze, kde jsou přenášené rámce šifrovány a pečetěny. Ustavovací a provozní fáze jsou stejné, jak jsme si je popsali u WPA3.

Na závěr této podkapitoly si můžeme shrnout veškeré zde popsané kryptografické protokoly pro Wi-Fi sítě (viz obr. 2.28). Pro ověřovací, resp. sjednávací fázi lze použít protokoly SAE, EAP, resp. OWE. Všechny tyto protokoly zajistí sjednání semena, přičemž protokoly SAE a EAP navíc zajišťují i autentizaci stran. U protokolu SAE je tato autentizace lokální a u protokolu EAP je centralizovaná. Protokol 4WH je využit k ustavení klíče a integrované symetrické kryptosystémy TKIP, CCMP nebo GCMP se využívají k šifrování a pečetění rámců v provozní fázi.



Obrázek 2.28: Vztahy mezi kryptografickými protokoly Wi-Fi sítí

2.7.2 Protokol MACsec

Protokol MACsec („Media Access Control Security“) [18] je kryptografický protokol, který je určen k zabezpečení komunikace v tzv. ethernetových spojích, tj. jedná se o kryptografický protokol linkové vrstvy. Ethernetové spoje jsou definovány pomocí standardů rodiny IEEE 802.3 a lze je stručně charakterizovat jako spoje, v nichž jsou linkové rámce přenášeny metalickými nebo optickými kabely. Tím se liší od Wi-Fi spojů (rodina standardů IEEE 802.11), kde se k přenosu rámců používají rádiové spoje.

Ethernetové spoje se používají k budování ethernetových sítí. Základním prvkem tohoto typu sítí je tzv. přepínač („switch“), což je síťové zařízení s více porty (rozhraními k dalším zařízením). K těmto portům se připojují počítače a jiné přepínače dané sítě. Přepínač pak podle adres v záhlaví ethernetových rámců tyto rámce mezi svými porty předává. Typické uspořádání ethernetové sítě využívající protokol MACsec ilustruje obr. 2.29.

Obrázek 2.29: Typické uspořádání ethernetové sítě

Na tomto obrázku vidíme dva přepínače R_1 a R_2, uživatelský počítač P připojený k přepínači R_1 a server S připojený k přepínači R_2. Spoje mezi zařízeními jsou ethernetové a předpokládejme, že jsou všechny zabezpečeny protokolem MACsec. Na každém spoji jsou k tomu mezi sousedními zařízeními ustaveny klíče *KH*. Podle našeho obrázku je mezi počítačem P a přepínačem R_1 ustaven klíč KH_1, mezi přepínačem R_1 a R_2 je ustaven klíč KH_2 a mezi přepínačem R_2 a serverem S je ustaven klíč KH_3.

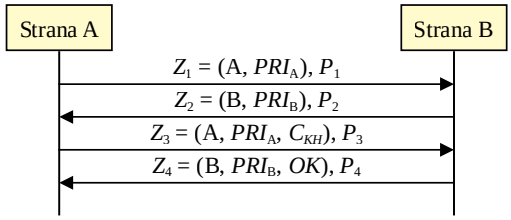
Pokud má počítač P odeslat paket serveru S, tak vloží daný paket do standardního ethernetového rámce, ten zabezpečí integrovaným symetrickým kryptosystémem pomocí klíče KH_1 a vytvořený MACsec rámec zašle přepínači R_1. V přijímacím portu přepínače R_1 je rámec ověřen, dešifrován a v podobě standardního rámce předán řídící jednotce přepínače. Ta podle cílové adresy v záhlaví rámce určí, že rámec má být předán do portu, který je ethernetovým spojem připojen k přepínači R_2. V určeném portu je rámec znovu zabezpečen pomocí klíče KH_2 a v podobě MACsec rámce je odeslán přepínači R_2. V přijímacím portu přepínače R_2 je tento rámec ověřen, dešifrován a ve standardní podobě předán řídící jednotce přepínače. Ta opět podle cílové adresy v záhlaví rámce

určí, že rámec má být předán do portu, který je připojen k serveru S. V tomto portu je rámec zabezpečen pomocí klíče KH_3 a odeslán serveru S, který přijatý MACsec rámec ověří a dešifruje. Z těla vzniklého standardního rámce vyjme paket a předá jej síťové vrstvě serveru.

Protokol MACsec zajišťuje důvěrnost i autentičnost dat v těle rámce a také autentičnost záhlaví rámce, přičemž k šifrování a k pečetění je standardizován symetrický integrovaný kryptosystém GCMP. Ten již obecně známe z předešlé kapitoly, a tak si jen popíšeme protokol pro ustavovací fázi, který je definován ve standardu [19]. Uvedený protokol nese zkratku MKA („MACsec Key Agreement Protocol") a jeho cílem je ustavit provozní klíč *KH* pro GCMP kryptosystém.

Předpokladem pro fungování protokolu MKA je takový stav, kdy obě strany ethernetového spoje (označme je A a B) disponují společným semenem *S* („Secure Connectivity Association Key" – *CAK*). Toto semeno je buď do obou zařízení vloženo jejich správcem (režim PSK – „Pre-Shared Key"), nebo je v průběhu centralizované autentizace uživatelských počítačů odvozeno z hodnoty *MSK*. Z předešlého víme, že hodnota *MSK* je výstupem autentizačního protokolu EAP – příklad jsme viděli u centralizované autentizace při popisu protokolu WPA3. Z hodnoty semene *S* mohou obě strany odvodit jak autentizační klíč *KA*, tak i transportní klíč *KT*. Platí, že $KA = \mathrm{ODF}(S, T_1)$ a $KT = \mathrm{ODF}(S, T_2)$, kde T_1 a T_2 jsou standardem definované kontexty.

Výměnu zpráv protokolu MKA ilustruje obr. 2.30. Dejme tomu, že protokol zahájí strana A vysláním zprávy $Z_1 = (\mathrm{A}, PRI_\mathrm{A})$, kde A je identifikátor odesílajícího zařízení a PRI_A je tzv. priorita zařízení A. Priority jednotlivých zařízení stanovuje při jejich konfiguraci ručně správce sítě, přičemž platí, že v provozní fázi generuje klíče *KH* zařízení s vyšší prioritou. Pečeť $P_1 = \mathrm{PCT}(Z_1, KA)$ je pak pečeť zprávy Z_1 vytvořená pomocí autentizačního klíče *KA*. Strana B pomocí autentizačního klíče *KA* nejprve ověří správnost pečeti P_1. Pokud $W = \mathrm{VER}(Z_1, P_1, KA) = 1$, tak je zpráva autentická, tj. strana A zná klíč *KA* a tedy i semeno *S*. Následně strana B porovná svoji prioritu PRI_B s prioritou PRI_A protistrany. Předpokládejme, že strana A má vyšší prioritu, tj. provozní klíče bude generovat strana A.



Obrázek 2.30: Výměna zpráv u protokolu MKA

V takovémto případě strana B odešle zprávu $Z_2 = (\mathrm{B}, PRI_\mathrm{B})$ spolu s pečetí P_2, kde B je identifikátor strany B a pečeť $P_2 = \mathrm{PCT}(Z_2, KA)$. Strana A ověří správnost pečeti P_2 a z hodnoty PRI_B

zjistí, že provozní klíče bude generovat ona. Vygeneruje proto první hodnotu provozního klíče *KH* a pomocí transportního klíče *KT* jej zašifruje do podoby kryptogramu C_{KH} = ENC(*KH*, *KT*). Dále sestrojí zprávu Z_3 = (A, PRI_B, C_{KH}), odvodí pro ni pečeť P_3 = PCT(Z_3, *KA*) a dvojici (Z_3, P_3) odešle protistraně.

Strana B ověří správnost pečeti P_3 a dešifruje kryptogram C_{KH}, čímž získá klíč *KH* = DEC(C_{KH}, *KT*). Následně straně A odešle zprávu Z_4 = (B, PRI_B, *OK*), kde *OK* je hlášení, že strana B přechází do provozní fáze. Zprávu Z_4 strana B doplní pečetí P_4 a tuto dvojici odešle straně A. Strana A pečeť ověří a na základě hlášení *OK* rovněž přejde do provozní fáze. Od tohoto okamžiku si obě strany vyměňují datové rámce, které jsou šifrovány a pečetěny symetrickým integrovaným kryptosystémem GCMP pomocí klíče *KH*. V případě potřeby nové hodnoty klíče *KH* se protokol MKA zopakuje.

3 Přístupové systémy

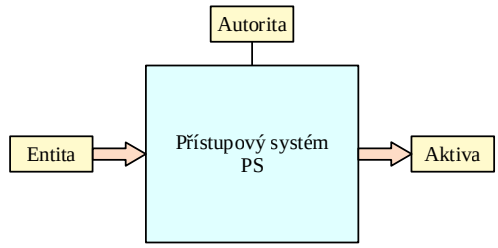
3 Přístupové systémy

Přístupové systémy jsou systémy, které regulují přístup k aktivům, přičemž pojem aktiva reprezentuje cokoliv, co je cenné. V praxi se zpravidla jedná o data, komunikační a počítačové služby, utajované skutečnosti či movitý majetek. O přístup k aktivům usilují tzv. entity, což jsou nejčastěji osoby, ale mohou to být i zařízení, jako jsou servery, robotická vozidla apod.

V této kapitole si zprvu vysvětlíme architekturu a způsob fungování přístupových systémů. Poté si popíšeme problematiku autentizace entit a následně také nejčastější autentizační protokoly (webové autentizace BAA a DAA a dále protokoly EAP, Kerberos a OpenID Connect). Poté si vysvětlíme autorizační protokol OAuth a nakonec si popíšeme přístupový protokol RADIUS spolu se systémy elektronické kontroly vstupu.

3.1 Architektura přístupových systémů

Jak již bylo uvedeno, přístupové systémy regulují přístup entit k aktivům. Zmíněná regulace spočívá v tom, že přístupový systém všem entitám v přístupu k chráněným aktivům všeobecně zabraňuje. Pokud však konkrétní entita splní stanovené podmínky, tak jí je přístup umožněn. Přístupový systém PS tak můžeme prakticky považovat za výběrově fungující překážku mezi entitami a aktivy (viz obr. 3.1).



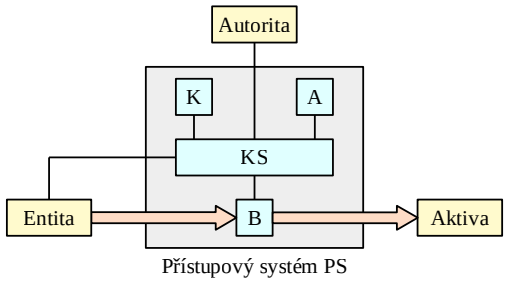
Obrázek 3.1: Princip přístupového systému

Entita se zájmem o přístup k aktivům musí nejprve zkontaktovat příslušnou autoritu (zpravidla se jedná o majitele či správce aktiv) a požádat ji o přístupová práva. Pokud entita splní stanovené požadavky, tak jí autorita příslušná práva *PR* udělí (tzv. autorizace). V klasických přístupových systémech jsou práva vázána na identitu entity, a tak se v rámci autorizace také sjednává:

- Identita *ID*, což bude v přístupovém systému PS unikátní identifikátor dané entity.
- Dokazovací faktor *DF*, což je něco, čím entita bude systému prokazovat svoji identitu a tedy i svá práva.
- Ověřovací faktor *OF*, což jsou data, jimiž bude systém identitu entity ověřovat.

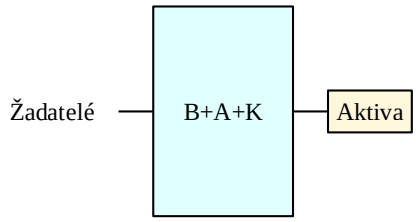
Pro autorizovanou entitu pak autorita zadá do přístupového systému její identitu *ID*, přidělená práva *PR* a ověřovací faktor *OF*. Od tohoto okamžiku může daná entita přistupovat k aktivům. Kromě samotného řízení přístupu může přístupový systém také evidovat, jaké akce entity v přístupovém systému provádějí a tyto informace pak autoritě poskytovat (tzv. účtování – „accounting“).

Obecnou strukturu přístupového systému ilustruje obr. 3.2. Jeho základními prvky jsou kontrolér K, autentizátor A a brána B. Pokud jsou uvedené prvky geograficky rozptýleny, tak je k jejich vzájemné komunikaci ještě nutný komunikační systém KS. Zmíněný komunikační systém také slouží k případné komunikaci prvků přístupového systému s entitami a s autoritou. V rámci komunikace mezi přístupovým systémem a entitou, která žádá o přístup (dále žadatel Z), se nejprve zjišťuje a ověřuje identita ID_Z žadatele. Kritériem je, že žadatel s identifikátorem ID_Z musí disponovat dokazovacím faktorem DF_Z. Tato fáze přístupu se nazývá autentizace a provádí ji autentizátor A. Pokud je žadatelova identita ID_Z potvrzena, tak toto zjištění autentizátor předá kontroléru K. Ten z údajů vložených autoritou zjistí, jaká práva PR_Z autorita žadateli Z udělila. Pokud z nich vyplývá, že žadatel má právo k požadovaným aktivům přistupovat, tak kontrolér zašle příkaz bráně B, aby žadateli Z přístup umožnila. Brána je obvykle buď zařízení pro přenos dat (pokud jsou aktivy například data a počítačové služby), nebo elektricky ovládaná výplň vstupního otvoru (např. dveře), pokud mají aktiva hmotný charakter.



Obrázek 3.2: Struktura přístupového systému

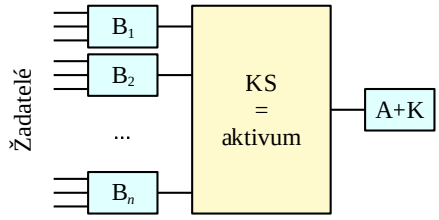
Podle míry integrace jednotlivých prvků přístupového systému lze rozeznávat různé konfigurace těchto systémů. V praxi jsou nejčastější konfigurace, které nazveme kompaktní a centralizovaná. Přístupový systém v kompaktní konfiguraci (viz obr. 3.3) tvoří prakticky jediné zařízení, které v sobě integruje všechny prvky přístupového systému – tj. plní úlohy kontroléru, autentizátora i brány. Typickým představitelem popsané konfigurace jsou webové servery. V tomto případě se uživatel Z připojí k serveru, v přihlašovací stránce zadá své jméno ID_Z a vůči autentizátoru se autentizuje svým heslem, tj. dokazovacím faktorem DF_Z. V případě úspěšné autentizace pak ta část programu serveru, která plní roli kontroléru, zjistí z databáze práva PR_Z. V souladu s těmito právy pak příslušná část programu serveru (brána) dovolí prohlížení stránek, k nimž má uživatel Z oprávnění. Uživatel tak získá přístup k vybraným stránkám webového serveru (tj. k aktivům).



Obrázek 3.3: Kompaktní konfigurace přístupového systému

Přístupový systém v centralizované konfiguraci se využívá ve scénářích s mnoha branami (viz obr. 3.4). V těchto případech jsou branami nejčastěji přístupové přepínače ethernetové sítě a aktivem jsou přenosové služby této sítě. Centralizovaná konfigurace spočívá v tom, že všechny brány B_1 až B_n jsou řízeny jediným kontrolérem K, který je spolu s centrálním autentizátorem A integrován do jednoho zařízení. Příkladem popsané konfigurace jsou systémy s protokolem RADIUS.

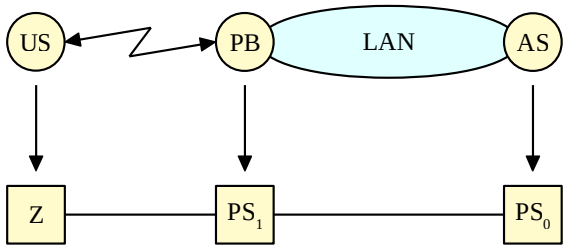
Trendem vývoje přístupových systémů jsou vícestupňové systémy. Uvedené systémy sestávají z několika kompaktních přístupových systémů, které jsou hierarchicky uspořádány. Platí přitom, že nadřízený přístupový systém řídí hierarchicky podřízený systém, přičemž chráněným aktivem nadřízeného systému je dokazovací faktor pro přístup k aktivům podřízeného systému. Uvedený koncept nabízí vyšší úroveň bezpečnosti (každé zařízení je autonomní přístupový systém), lepší škálovatelnost a také možnost kooperace přístupových systémů různých autorit.



Obrázek 3.4: Centralizovaná konfigurace přístupového systému

S příkladem vícestupňového uspořádání jsme se již setkali u komplexu WPA (viz obr. 3.5). U něho se konkrétně jedná o dvoustupňový přístupový systém, kde nadřízený, resp. podřízený přístupový systém označíme PS_0, resp. PS_1. Systém PS_0 je integrován do autentizačního serveru AS a systém PS_1 je přístupovým bodem PB do Wi-Fi sítě. Uživatelská stanice US (žadatel Z) nejprve přes přístupový bod PB (tj. přes PS_1) a přes síť LAN komunikuje s PS_0, který stanovenou autentizační metodou ověří, zda žadatel disponuje dokazovacím faktorem DF_0. Jak již víme, v rámci zmíněné autentizace je také ustavena tajná hodnota *PMK*, která je v této souvislosti aktivem přístupového

systému PS_0. Plní totiž úlohu dokazovacího faktoru DF_1 pro autentizaci v podřízeném přístupovém systému PS_1. Přístupový systém PS_0 poté odešle příkaz podřízenému přístupovému systému PS_1, v němž se stanovuje, že pokud žadatel Z prokáže znalost hodnoty *PMK*, tak má být připojen do Wi-Fi sítě. Přístupový systém PS_1 tuto skutečnost ověří a v kladném případě uživatelskou stanici do sítě připojí.



Obrázek 3.5: WPA3 jako dvoustupňový přístupový systém

Ve vícestupňových přístupových systémech někdy nacházíme dílčí přístupové systémy, které nemají autentizátor. Takovéto přístupové systémy existují ve dvou variantách. U první varianty nejsou práva vázána na identitu, nýbrž přímo na dokazovací faktor žadatele. Žadatel v tomto případě prokáže kontroléru vlastnictví dokazovacího faktoru (obvykle mu žadatel tuto tajnou hodnotu zašle) a kontrolér mu na tomto základě umožní přístup k aktivu. S příkladem popsaného řešení se potkáme u protokolu OAuth. Druhou variantou je postup, kdy jsou aktiva zpřístupněná žadateli zašifrována tajným klíčem K_Z, který je zároveň dokazovacím faktorem DF_Z žadatele. Žadatel Z v tomto případě udá kontroléru svoji identitu ID_Z a ten mu žádaná aktiva zpřístupní jako kryptogram zašifrovaný klíčem K_Z. S touto variantou se potkáme u protokolu Kerberos.

3.2 Autentizace

Ke svému jednoznačnému odlišení se entity označují unikátními identifikátory *ID* (např. v případě osob jde typicky o křestní jména a příjmení). Pojem identifikace pak označuje proces zjištění identifikátoru konkrétní entity, přičemž tato identifikace může být průkazná, nebo neprůkazná. V případě průkazné identifikace (tzv. autentizace) je identita dané entity zjištěna s požadovanou mírou záruk. V případě neprůkazné identifikace nelze zjištěný identifikátor entity považovat za dostatečně věrohodný. Typicky se jedná o situaci, kdy se například osoba jednoduše představí (tj. uvede svůj identifikátor) bez toho, že by své tvrzení nějak prokázala.

V dalším si autentizaci budeme vysvětlovat pro případ, kdy entitou bude osoba. Tento případ je jednak názornější a také obecnější. Již jsme zmínili, že v průběhu autorizace je osobě Z přidělen unikátní identifikátor ID_Z a pro tuto osobu jsou sjednány její dokazovací faktor DF_Z a ověřovací

faktor OF_Z. Dokazovací faktor je něco, co je unikátní a pro případného útočníka prakticky nezískatelné. Příkladem je specifická struktura papilárních linií prstu osoby nebo znalost tajné informace. Ověřovacím faktorem OF_Z pak jsou data, jimiž lze ověřit, že kontrolovaná osoba disponuje dokazovacím faktorem DF_Z. Pro výše uvedené příklady je ověřovacím faktorem obrázek papilárních linií prstu, resp. znalost tajné informace.

Obecně vzato, dokazovacím faktorem mohou být buď unikátní a nepadělatelné rysy, nebo unikátní a tajná data. Nosičem těchto rysů, resp. dat pak může být buď samotná osoba, nebo předmět, jímž tato osoba disponuje. Podle typu dokazovacího faktoru (tj. rysy versus data) a podle typu nosiče dokazovacího faktoru (tj. osoba versus předmět) můžeme autentizaci klasifikovat do čtyř tříd podle obr. 3.6.

Třídy autentizace		Dokazovací faktor *DF*	
		Rysy	Data
Nosič *DF*	Osoba	Biometrika	Heslo
	Předmět	Průkaz	Hardware

Obrázek 3.6: Třídy autentizace

V případě autentizace biometrikou je nosičem *DF* samotná osoba, přičemž dokazovacím faktorem je tzv. biometrika, což jsou změřené a číselně vyjádřené biologické nebo behaviorální (tj. týkající se chování) znaky osoby. Typicky bývá dokazovacím faktorem uspořádání papilárních linií prstů nebo skvrn na oční duhovce. Znaky osoby jsou při její autorizaci změřeny a uloženy do souboru, který slouží jako ověřovací faktor *OF*. Při autentizaci osoby se pak její biometrika změří znovu, a pokud jsou výsledky tohoto měření v dostatečné shodě s *OF*, tak je autentizace úspěšná. S biometrickou autentizací se setkáváme zejména u přístupových systémů do místností, budov a areálů. Často se používá také při autentizaci osob vůči přenosným elektronickým zařízením, jako jsou notebooky nebo smartfony. U této třídy autentizace se kryptografie prakticky nepoužívá, a tak se ji v dalším již věnovat nebudeme.

U autentizace heslem je nosičem *DF* opět osoba, ale dokazovacím faktorem jsou tajná data uložená v paměti osoby. Typicky se jedná o heslo či PIN („Personal Identification Number“). Pokud tato tajná data osoby Z označíme jako PSW_Z, tak dokazovací faktor $DF_Z = PSW_Z$. Pro ověřovací faktor obvykle platí, že $OF_Z = DF_Z = PSW_Z$ a pak kritériem pro úspěšnou autentizaci je, že dokazovací data DD_Z (data předaná žadatelem autentizátoru) jsou stejná jako ověřovací faktor, tj. $DD_Z = OF_Z$. Jinou možností ověření hesla je varianta, kdy $OF_Z = \mathrm{HSF}(PSW_Z)$. Kritériem úspěšné autentizace pak je, že žadatel autentizátoru předá $DD_Z = PSW_Z$ a autentizátor zkontroluje, zda $\mathrm{HSF}(DD_Z) = OF_Z$. Výhodou této varianty je vyšší bezpečnost. Pokud se útočník dostane k seznamu ověřovacích faktorů v autentizátoru, tak vzhledem k jednosměrnosti hešovací funkce HSF

není schopen z hodnot OF_Z = HSF(PSW_Z) určit hodnotu hesel PSW_Z. U autentizace heslem je důležité, že pokud přenosový kanál mezi žadatelem a autentizátorem není bezpečný (typicky přihlašování ke vzdálenému serveru), tak heslo musí být přenášeno v zašifrované podobě.

V případě autentizace průkazem je nosičem *DF* předmět a dokazovacím faktorem jsou unikátní a nepadělatelné rysy tohoto předmětu. V praxi je průkaz obvykle papírová či plastová karta, která je proti padělání chráněna různými ochrannými prvky (např. hologramové nálepky, soutisky apod.). Zároveň je v průkazu uveden také identifikátor osoby. Autentizátor si pomocí ochranných prvků průkazu nejprve ověří, zda je průkaz vydán příslušnou autoritou a není pozměněn. V kladném případě potom musí být autentický i identifikátor osoby ID_Z uvedený na průkazu – držitel průkazu je pak považován za osobu Z. Autentizace průkazem je typická pro scénář, kdy autentizátorem je osoba (např. úředník či policista). V technických systémech se tento typ autentizace nepoužívá, protože snímače potřebné k ověřování ochranných prvků průkazu jsou drahé.

Z kryptografického hlediska je pro nás nejzajímavější třída autentizace hardwarem. V tomto případě je nosičem *DF* vhodný hardware (typicky čipová karta) a dokazovacím faktorem pak jsou tajná data uložená v tomto hardware. Autentizace se liší podle toho, zda hardware je paměťové úložiště, nebo výpočetní zařízení. V případě paměťového úložiště (typicky karta s magnetickým proužkem) je v úložišti jako dokazovací faktor zapsán tajný identifikátor osoby ID_Z. Ověřovacím faktorem je rovněž ID_Z, a tak kritériem úspěšné autentizace je shoda ID_Z v paměťovém úložišti hardwaru s ověřovacím faktorem.

Pokud má autentizační hardware podobu výpočetního zařízení (typicky mikroprocesorová karta), tak se k autentizaci využívají kryptografické techniky. Nejčastější variantou je v takovémto případě autentizační protokol, kdy dokazovacím faktorem DF_Z osoby Z je tajný pečetící klíč K_Z, resp. soukromý podepisovací klíč SK_Z. Ověřovacím faktorem OF_Z pak je tajný ověřovací klíč K_Z, resp. veřejný ověřovací klíč VK_Z. Autentizátor nejprve hardwaru osoby Z zašle náhodnou výzvu N. Ten na ni odpoví dokazovacími daty DD = PCT(N, K_Z), resp. DD = PCT(N, SK_Z). Autentizátor tuto odpověď ověří, tj. vypočítá autentizační indikátor W = VER(N, DD, K_Z), resp. VER(N, DD, VK_Z). Pokud W = 1, tak kontrolovaná osoba disponuje hardwarem s dokazovacím faktorem DF_Z osoby Z a tudíž se předpokládá, že se jedná o osobu Z.

Na závěr této podkapitoly je ještě vhodné zmínit, že autentizátor může autentizaci provést se znalostí i bez znalosti identifikátoru osoby. První případ nazveme prezenční autentizací. V tomto případě se žadatel autentizátoru představí, tj. uvede svůj identifikátor ID_Z. Autentizátor pak ví, že k ověření identity žadatele má ze své databáze ověřovacích faktorů použít faktor OF_Z. Druhý případ nazveme prohledávací autentizací. Žadatel se v tomto případě nepředstavuje a autentizátor pro žadatelem předložená dokazovací data DD postupně zkouší, zda vyhoví pro některý z ověřovacích faktorů. Pokud DD vyhoví pro OF_Z, tak se předpokládá, že žadatelem je osoba Z. Prezenční autentizace je efektivnější a bezpečnější, a proto se prohledávací autentizace používá jen ve scénářích, kdy by zadávání identifikátoru osoby autentizaci neúměrně zdržovalo. Typicky se jedná o biometrickou nebo heslovou autentizaci v přístupových systémech do budov. Vyzkoušení až

tisíců ověřovacích faktorů pro aktuálně sejmutou biometriku, nebo vložený PIN je totiž mnohem rychlejší než uživatelovo vyťukání vlastního identifikátoru na klávesnici terminálu.

3.3 Autentizační protokoly

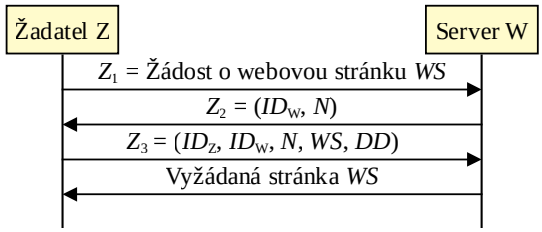
Protokoly, s nimiž se v přístupových systémech setkáme, rozdělíme na autentizační, autorizační a přístupové. Autentizačními protokoly budeme rozumět protokoly, jejichž jediným účelem je autentizace žadatele. Z této kategorie protokolů si popíšeme HTTP autentizaci BAA a DAA, obecný protokol EAP, síťový protokol Kerberos a webový protokol OpenID Connect. Druhou kategorii nazveme autorizační, protože jsou určeny k autorizaci žadatelů. Jediným zástupcem této kategorie protokolů bude protokol OAuth. Poslední podkapitolou budou přístupové protokoly, které umožňují řídit přístup žadatelů k aktivům. Z této kategorie protokolů si vysvětlíme protokol RADIUS, který slouží k řízení přístupu uživatelských počítačů do sítě. Navíc si zde vysvětlíme i systémy k řízení přístupu osob do místností a budov – tzv. systémy elektronické kontroly vstupu.

3.3.1 Autentizace BAA a DAA

Autentizace BAA a DAA slouží k autentizaci stran, které komunikují prostřednictvím protokolu HTTP. Tento protokol je základem webového (z anglického „World Wide Web“ – WWW) systému, což je globální systém pro práci s elektronickými dokumenty psanými v jazyce HTML („Hypertext Markup Language“). Uvedené dokumenty (neboli webové stránky) typicky obsahují texty, obrázky, videa a také odkazy na jiné stránky. Webové stránky jsou ukládány na webové servery, z nich si je zájemci (dále uživatelé) prostřednictvím protokolu HTTP přenášejí na svůj počítač a na něm si pak stránky prohlížejí pomocí specializované aplikace, která se nazývá webový prohlížeč.

K autentizaci uživatelů webových serverů se používá autentizace heslem. Podle normy [20] jsou standardizovány dvě metody, které se označují zkratkami BAA („Basic access authentication“) a DAA („Digest access authentication“). U metody BAA se heslo uživatele jednoduše přenáší protokolem HTTP v otevřené podobě, a proto je nutné spojení mezi uživatelem a serverem nejprve zabezpečit – obvykle se používá protokol TLS. Metoda DAA využívá kryptografii přímo, a proto si ji zde popíšeme, přičemž se omezíme na nejjednodušší variantu.

Webový server W označíme identifikátorem ID_W a uživatele Z identifikátorem ID_Z. Dokazovacím faktorem DF_Z uživatele Z je heslo PSW_Z a jeho ověřovacím faktorem je heš řetězce z ID_Z, ID_W a hesla, tj. $OF_Z = \mathrm{HSF}(ID_Z \parallel ID_W \parallel PSW_Z)$. Autentizační protokol DAA ilustruje obr. 3.7. Žadatel Z nejprve požádá zprávou Z_1 o zaslání webové stránky *WS*. Server mu zašle zprávu Z_2 se svým identifikátorem ID_W a výzvou, která je náhodným číslem *N*. Uživatel poté vypočítá dokazovací data $DD = \mathrm{HSF}(X \parallel N \parallel Y)$, kde $X = \mathrm{HSF}(ID_Z \parallel ID_W \parallel PSW_Z)$ a $Y = \mathrm{HSF}(Z_1)$. Serveru poté zašle zprávu Z_3, která obsahuje identifikátory ID_Z a ID_W, výzvu *N*, označení požadované webové stránky *WS* a dokazovací data *DD*.



Obrázek 3.7: Autentizační protokol DAA

Ze vztahu pro řetězec X vidíme, že $X = OF_Z$, a tak webový server může vypočítat kontrolní hodnotu $DD' = \mathrm{HSF}(OF_Z \| N \| Y)$. Pokud platí, že přijatá dokazovací data DD jsou rovna kontrolní hodnotě DD', tak webový server má záruku, že protistrana zná heslo PSW_Z a jedná se tedy o uživatele s identifikátorem ID_Z. Pokud má žadatel Z právo si prohlížet stránku *WS*, tak mu ji server zašle.

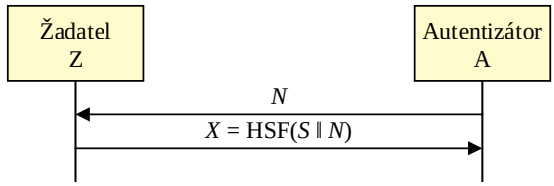
3.3.2 Protokol EAP

Protokol EAP („Extensible Authentication Protocol“) [21] není jeden konkrétní protokol, ale tzv. autentizační rámec. Tento rámec definuje několik obecných typů zpráv a definuje mechanismy výměny těchto zpráv. Konkretizací typů zpráv (typicky definicí podtypů zpráv a obsahu jejich těla) jím lze implementovat prakticky libovolnou metodu autentizace. V současné době je standardizováno více než čtyřicet různých metod, přičemž my se zde seznámíme se základní metodou EAP-MD5 a stručně si vysvětlíme metody EAP-TLS, EAP-TTLS a EAP-PEAP. V této souvislosti zmíníme i rozšíření EAPoL a seznámíme se s využitím autentizačních EAP metod podle standardu IEEE 802.1X.

Autentizace EAP-MD5 je velmi rozšířená metoda autentizace, i když v současné době už není příliš bezpečná. Je založena na tom, že jak žadatel Z, tak autentizátor A disponují tajnou hodnotou S. Pro dokazovací faktor DF a ověřovací faktor OF tedy platí, že $DF = OF = S$. Ze schématu protokolu na obr. 3.8 je zřejmé, že protokol zahajuje autentizátor odesláním náhodné výzvy N. Žadatel odpoví zprávou, v níž je uvedena hodnota $X = \mathrm{HSF}(S \| N)$, přičemž HSF je hešovací funkce MD-5. Autentizátor vypočítá svoji kontrolní hodnotu $X' = \mathrm{HSF}(S \| N)$, a pokud $X = X'$, tak má záruku, že protistrana disponuje tajnou hodnotou semena S a tudíž se jedná o žadatele Z.

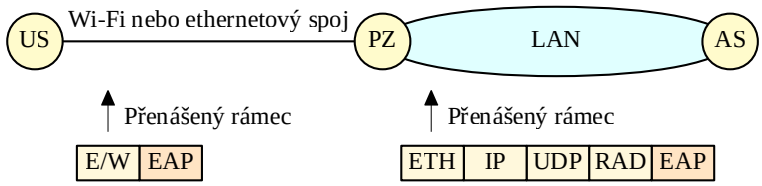
Výhodou protokolu EAP je jeho nezávislost na přenosovém protokolu. Zprávy EAP protokolu se proto mohou přenášet v protokolech různých vrstev IP architektury, přičemž v linkové vrstvě se EAP zprávy přenášejí zejména v ethernetových nebo Wi-Fi rámcích a v aplikační vrstvě se EAP zprávy nejčastěji přenášejí jako přílohy zpráv protokolu RADIUS (viz dále). Uvedená kombinace protokolů se využívá k řízení přístupu uživatelských stanic k Wi-Fi a LAN sítím podle standardu

IEEE 802.1X [22]. Postup podle uvedeného standardu jsme sice již zmínili u protokolu MACsec a komplexu WPA, ale zde si tuto problematiku vysvětlíme obecněji.



Obrázek 3.8: Autentizace EAP-MD5

Na obr. 3.9 vidíme typické uspořádání přístupového systému podle standardu IEEE 802.1X. Uživatelská stanice US (typicky počítač s Wi-Fi nebo ethernetovou síťovou kartou) se připojuje na přístupové zařízení PZ, které je buď přístupovým bodem Wi-Fi sítě, nebo přepínačem ethernetové sítě, který je vyčleněn pro připojování počítačů uživatelů. Ve spoji mezi US a PZ probíhá komunikace v linkové vrstvě pomocí linkových rámců. Na obrázku reprezentuje bitový blok E/W záhlaví ethernetového, resp. Wi-Fi rámce a v těle tohoto rámce vidíme EAP zprávu. Komunikace mezi zařízením PZ a serverem AS se odehrává pomocí zpráv protokolu RADIUS, což je protokol aplikační vrstvy. Na obrázku vidíme, že v tomto případě je EAP zpráva přílohou zprávy protokolu RADIUS (tato zpráva je označena jako blok RAD). V transportní vrstvě je celá zpráva protokolu RADIUS (tj. RAD+EAP) přenášena pomocí datagramu UDP protokolu (na obrázku je záhlaví datagramu označeno zkratkou UDP), přičemž tento datagram je v síťové vrstvě přenášen IP paketem (jeho záhlaví je označeno IP). V linkové vrstvě je IP paket nakonec přenášen v ethernetovém rámci (jeho záhlaví je označeno ETH). Popsané řešení dovoluje, aby uživatelská stanice US mohla pomocí zpráv protokolu EAP komunikovat nejen s přístupovým zařízením PZ, ale prostřednictvím PZ i s autentizačním serverem AS.



Obrázek 3.9: Řízení přístupu podle IEEE 802.1X

Po připojení uživatelské stanice US umožní přístupové zařízení PZ této stanici komunikaci jen s autentizačním serverem AS. K tomu, aby stanice US mohla využívat služeb sítě LAN, se totiž musí nejprve autentizovat, k čemuž se používají právě metody protokolu EAP. Pro nasazení na

spoji mezi US a PZ musel být tento protokol rozšířen o další zprávy, čímž vznikl rámec EAPoL („EAP over LAN) [22]. Například u popisu metody EAP-MD5 jsme viděli, že autentizaci vždy zahajuje autentizátor, což je v uspořádání podle obr. 3.9 dosti nepraktické, protože v protokolu EAP není definována zpráva, kterou by uživatelská stanice mohla dát přístupovému zařízení najevo, že se právě připojila. Rozšíření EAPoL právě takovouto zprávu spolu s řadou dalších zpráv (např. zprávou pro odhlášení US od PZ) definuje.

V rámci architektury podle IEEE 802.1X se ke vzájemné autentizaci mezi stanicí US a serverem AS používají EAP metody, které kromě autentizace obou stran umožňují ustavit i semeno *S*. Toto semeno poté AS šifrovaně přenese do PZ, kde je použito ke vzájemné autentizaci mezi US a PZ a k ustavení klíčů pro kryptografické zabezpečení datového provozu mezi těmito stranami. V případě kabelového připojení stanice US k zařízení PZ se používá protokol MACsec a v případě rádiového připojení se používá komplex WPA.

K autentizaci mezi stanicí US a serverem AS se často používají metody EAP-TLS, EAP-TTLS a PEAP. Všechny tyto metody jsou založeny na protokolu TLS. Princip je takový, že v tělech EAP zpráv jsou přenášeny datové bloky protokolu TLS. Obě strany si tímto způsobem mezi sebou vytvoří TLS spojení, přičemž v rámci zahajovací fáze protokolu TLS se autentizuje server AS pomocí svého veřejného klíče z certifikátu. Autentizace uživatelské stanice se liší podle použité metody. V případě metody EAP-TLS se stanice US rovněž autentizuje pomocí klíče ze svého certifikátu. Protože je však pravidelná výměna certifikátů u velkého počtu uživatelů složitá, tak se často používají metody EAP-TTLS („Tunneled TLS“) a PEAP („Protected EAP“). U těchto metod se uživatel autentizuje až v provozní fázi protokolu TLS. V případě EAP-TTLS se v EAP zprávách přenášejí bloky TLS protokolu, v nichž se při autentizaci žadatele šifrovaně přenáší heslo uživatele. Zprávy typu EAP-PEAP přenášejí rovněž bloky TLS protokolu, ale ty při autentizaci žadatele obsahují v zašifrované podobě opět EAP zprávy – v tomto případě však zprávy EAP metody, která je autoritou určena k autentizaci uživatelů.

Ustavení semena *S* v rámci výše uvedených metod je zajištěno v zahajovací fázi TLS protokolu, obvykle pomocí DH protokolu. Z tohoto semena se v různých EAP metodách různými odvozovacími funkcemi odvozuje tajná hodnota, která se podle standardu IEEE 802.1X nazývá *MSK* („Master Session Key“). Po skončení vzájemné autentizace předá server AS hodnotu *MSK* přístupovému zařízení PZ. K šifrovanému přenosu této hodnoty se obvykle používají zprávy definované ve standardech EAP, přičemž potřebné šifrovací klíče vkládá do serveru AS a zařízení PZ správce sítě. Jak se hodnota *MSK* používá k následné vzájemné autentizaci mezi stanicí US a zařízením PZ a k ustavení klíčů, jsme si již popsali v částech věnovaných protokolu MACsec a komplexu WPA3.

3.3.3 Protokol Kerberos

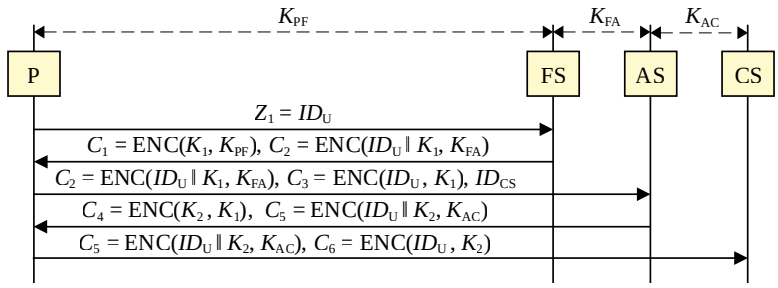
Protokol Kerberos [23] je síťový autentizační protokol, který uživatelům sítě velké organizace umožňuje tzv. jednorázové přihlášení („Single sign on“ – SSO). Jedná se o takové přihlášení, kdy

se uživatel U při příchodu do práce autentizuje vůči svému počítači P a tento počítač se pak po celý zbytek pracovní doby autentizuje vůči serverům dané organizace jménem svého uživatele. Z kryptografického hlediska je protokol Kerberos založen na symetrické kryptografii, přičemž tajné klíče K hrají roli jak dokazovacího tak i ověřovacího faktoru, tj. $K = DF = OF$.

Jednotlivé strany protokolu Kerberos budeme nazývat počítač P uživatele U, faktorový server FS („Authentication server"), akreditační server AS („Ticket Granting Service") a cílový server CS („Application server"). Princip je takový, že počítač P po úspěšné autentizaci uživatele U získá přístup k uživatelově dlouhodobému dokazovacímu faktoru, což je klíč K_{PF}. Následně si počítač vyžádá od faktorového serveru FS krátkodobý autentizační faktor, jímž je klíč K_1. S tímto klíčem se pak bude počítač P autentizovat vůči akreditačnímu serveru, přičemž platnost tohoto faktoru bývá obvykle pracovní doba jednoho dne. V případě, kdy uživatel U bude potřebovat nějakou službu (např. vytisknutí dokumentu), se počítač P autentizuje vůči akreditačnímu serveru AS a získá od něho jednorázový faktor, kterým je klíč K_2. Tímto klíčem se následně počítač autentizuje vůči cílovému serveru CS (v našem příkladu vůči tiskovému serveru). Pokud po úspěšné autentizaci zjistí přístupový kontrolér daného cílového serveru, že uživatel U má právo dokument vytisknout, tak mu dokument bude vytisknut. Skutečnost, že v protokolu jsou definovány dlouhodobý, krátkodobý a jednorázový klíč zvyšuje bezpečnost provozu.

Co se týká správy klíčů, tak každému počítači P je přidělen klíč K_{PF} jemu příslušného uživatele U. Faktorový server FS zná klíče K_{PF} všech uživatelů a zná i klíč K_{FA}, což je klíč pro šifrovaný přenos krátkodobých faktorů směrem k akreditačnímu serveru AS. Akreditační server pak kromě klíče K_{FA} zná ještě klíče K_{AC} všech cílových serverů. Ty se používají k šifrovanému přenosu jednorázových faktorů od akreditačního serveru k jednotlivým cílovým serverům. Fungování protokolu Kerberos si nyní můžeme vysvětlit podle obr. 3.10.

Uživatel U se po příchodu do práce autentizuje vůči svému počítači P heslem PSW_U. Pomocí tohoto hesla nyní může počítač získat i uživatelův dokazovací faktor K_{PF}. Ten je buď uložen v zašifrované podobě na disku počítače, přičemž potřebný dešifrovací klíč se vhodnou odvozovací funkcí odvozuje z hesla PSW_U, nebo se klíč K_{PF} odvozuje z hodnoty PSW_U přímo.



Obrázek 3.10: Schéma protokolu Kerberos

Počítač P pak zašle faktorovému serveru FS zprávu $Z_1 = ID_U$, což je fakticky žádost o přidělení krátkodobého dokazovacího faktoru pro uživatele U. Tento faktor obvykle platí pro pracovní dobu jednoho dne a má podobu tajného klíče K_1. Faktorový server zkontroluje, zda má ve svém seznamu uživatele s identifikátorem ID_U. V tomto seznamu rovněž nalezne i uživatelův klíč K_{PF}. Pokud je daný uživatel v seznamu uveden, tak faktorový server vygeneruje náhodný klíč K_1 a počítači P zašle přidělovací kryptogram $C_1 = \mathrm{ENC}(K_1, K_{PF})$ a potvrzovací kryptogram $C_2 = \mathrm{ENC}(ID_U \parallel K_1, K_{FA})$. Počítač P přidělovací kryptogram C_1 dešifruje, čímž získá dokazovací faktor $K_1 = \mathrm{DEC}(C_1, K_{PF})$ a potvrzovací kryptogram C_2 si uloží pro pozdější použití. Z rovnice pro kryptogram C_2 vidíme, že obsahuje ověřovací faktor K_1 i identifikátor ID_U, které jsou zašifrovány klíčem K_{FA}. Tento klíč sdílí spolu s faktorovým serverem FS akreditační server AS. Pro něho tak bude kryptogram C_2 potvrzení, že uživatel s identifikátorem ID_U obdržel od faktorového serveru FS dokazovací, resp. ověřovací faktor K_1.

Pokud bude chtít uživatel U v průběhu pracovní doby využít službu nějakého cílového serveru CS, tak se postupuje následovně. Počítač nejprve zašle akreditačnímu serveru AS potvrzovací kryptogram C_2, dokazovací kryptogram $C_3 = \mathrm{ENC}(ID_U, K_1)$ a nakonec ještě identifikátor cílového serveru ID_{CS}. Podle tohoto identifikátoru akreditační server zjistí, že žadatel má zájem o službu poskytovanou serverem CS. Následně akreditační server pomocí klíče K_{FA} dešifruje potvrzovací kryptogram C_2, čímž obdrží $ID_U \parallel K_1 = \mathrm{DEC}(C_2, K_{FA})$. Pro server AS je to důvěryhodné potvrzení od serveru FS, že uživatelovi s identifikátorem ID_U byl jako dokazovací faktor stanoven klíč K_1. Nakonec si akreditační server ověří, zda se skutečně jedná o žádost od uživatele s ID_U. Musí platit, že dešifrováním dokazovacího kryptogramu C_3 získá server AS stejný identifikátor, jako je uveden v potvrzovacím kryptogramu C_2. Pokud tedy platí, že $\mathrm{DEC}(C_3, K_1) = ID_U$, tak potom akreditační server vygeneruje pro uživatele U jednorázový dokazovací faktor, kterým je klíč K_2. Následně zašle počítači P přidělovací kryptogram $C_4 = \mathrm{ENC}(K_2, K_1)$ a potvrzovací kryptogram $C_5 = \mathrm{ENC}(ID_U \parallel K_2, K_{AC})$.

Uživatel dešifrováním přidělovacího kryptogramu C_4 zjistí jemu přidělený jednorázový faktor $K_2 = \mathrm{DEC}(C_4, K_1)$. Následně cílovému serveru CS přepošle přijatý potvrzovací kryptogram C_5 a k němu připojí svůj dokazovací kryptogram $C_6 = \mathrm{ENC}(ID_U, K_2)$. Cílový server nejprve dešifruje pomocí klíče K_{AC} kryptogram C_5, a tak obdrží $\mathrm{DEC}(C_5, K_{AC}) = ID_U \parallel K_2$. Je to pro něj potvrzení od akreditačního serveru, že uživateli s identifikátorem ID_U byl stanoven jako dokazovací faktor klíč K_2. Poté tímto klíčem dešifruje dokazovací kryptogram C_6 a měl by obdržet stejný identifikátor, který byl uveden v dešifrovaném kryptogramu C_5. Pokud tedy $\mathrm{DEC}(C_6, K_2) = ID_U$, tak žadatelem o jím poskytovanou službu je skutečně žadatel U. Tím protokol Kerberos končí. Autentizátor cílového serveru předá své zjištění svému kontroléru a ten zjistí práva uživatele s identifikátorem ID_U. Pokud požadavek uživatele zjištěným právům vyhovuje, tak je uživateli požadovaná služba poskytnuta.

Faktorový a akreditační server by teoreticky mohly být jediným serverem a skutečně obvykle oba běží na jednom počítači. Jejich logické oddělení však umožňuje autentizační spolupráci různých sítí. Máme-li třeba síť A se servery FS_A a AS_A a dále pak síť B se servery FS_B a AS_B, tak pokud mezi FS_A a AS_B, resp. FS_B a AS_A ustaví správci těchto sítí potřebné šifrovací klíče, tak potom

mohou například uživatelé sítě A s krátkodobým faktorem od FS_A získávat jednorázové faktory od AS_B, čímž mohou využívat služeb serverů sítě B.

Z pohledu přístupových systémů lze výše popsanou infrastrukturu považovat za třístupňový přístupový systém, v němž protokol Kerberos zajišťuje autentizaci. Hierarchicky nejvyšší je faktorový server FS, kde dokazovacím faktorem je uživatelův klíč K_{PF} a aktivem je krátkodobý dokazovací faktor K_1. Povšimněme si, že v tomto případě je autentizace implicitní, tj. systém je bez autentizátoru. K aktivu K_1 v přidělovacím kryptogramu C_1 se totiž nakonec dostane jen ten, kdo zná dokazovací faktor uživatele, jímž je klíč K_{PF}. Druhým systémem je akreditační server AS. Dokazovacím faktorem je v tomto případě klíč K_1 a aktivem je jednorázový dokazovací faktor K_2. Posledním přístupovým systémem je cílový server CS. Zde je dokazovacím faktorem klíč K_2 a aktivem je služba tímto serverem poskytovaná.

Za poznámku rovněž stojí přenos dokazovacích, resp. ověřovacích faktorů K_1 a K_2. Server, který faktor vydává (tj. FS, resp. AS), jej vydává v šifrované podobě. Dokazovací faktor uživatele vůči podřízenému systému je šifrován dokazovacím faktorem uživatele vůči vydávajícímu systému (tyto kryptogramy jsme nazvali přidělovací) a ověřovací faktor uživatele je šifrován klíčem, který je ustaven mezi nadřízeným a jemu podřízeným systémem (tyto kryptogramy jsme nazvali potvrzovací). Všimněme si také, že kryptogram s ověřovacím faktorem uživatele není zasílán podřízenému systému přímo, nýbrž mu je zasílán přes žadatele. Není tak zapotřebí, aby se v protokolu řešily přenosy mezi servery.

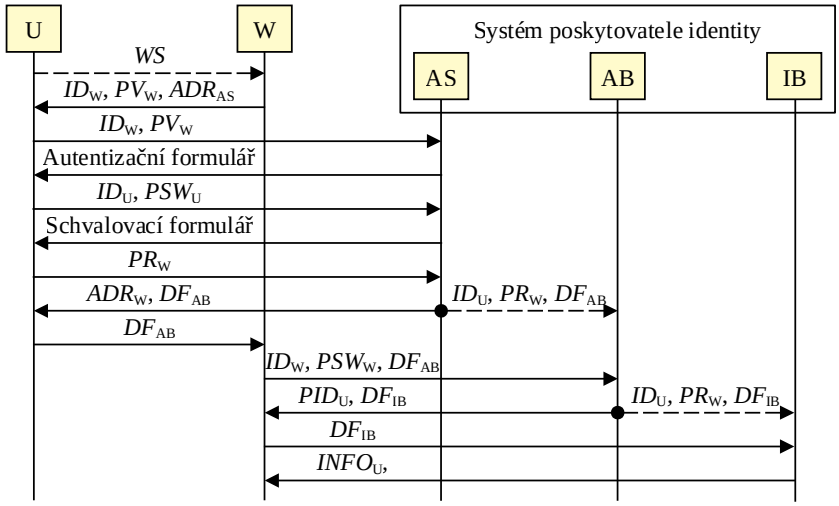
3.3.4 Protokol OpenID Connect

Protokol OpenID Connect je protokol, který je určen pro externí autentizaci uživatelů webových serverů. Provozovatelé webových serverů tak nemusí provozovat vlastní autentizátor a využívají externí autentizační službu tzv. poskytovatele identity. Tato služba je výhodná i pro uživatele, protože si pro *n* různých webových serverů nemusí pamatovat *n* různých identit a jim příslušných hesel.

Protokol OpenID Connect je definován standardem [24] a nahrazuje svého předchůdce, kterým byl protokol OpenID. Ve standardu je popsáno několik variant protokolu, ale my si zde uvedeme jen nejčastěji používanou variantu s tzv. autorizačním kódem. Tato varianta je uvedena na obr. 3.11. Stranami protokolu je zájemce o webové stránky (dále uživatel U), webový server W a systém poskytovatele identity, který sestává z autentizačního serveru AS, autentizační brány AB a informační brány IB. Z pohledu přístupových systémů jsou webové stránky serveru W aktivy, uživatel U vystupuje v roli žadatele a webový server W plní roli jak kontroléru K, tak i brány B. Roli autentizátoru A hraje systém poskytovatele identity.

Skutečnost, že autentizátor v protokolu OpenID Connect sestává namísto jednoho z celkem tří serverů vyplývá z toho, že protokol OpenID Connect je nadstavbou protokolu OAuth (viz dále). V uvedeném protokolu jsou definovány tři servery, které z pohledu webového serveru W fakticky tvoří třístupňový přístupový systém, jehož finálním aktivem je potvrzení o identitě uživatele U.

V souladu s tímto pohledem je server AS pro server W vrcholovým přístupovým systémem PS_{AS}. Tento server nejprve autentizuje uživatele U a poté serveru W vydává dokazovací faktor DF_{AB}, což je zároveň aktivum přístupového systému PS_{AS}. Webový server pomocí dokazovacího faktoru DF_{AB} následně získá od serveru AB (přístupový systém PS_{AB}) aktivum tohoto přístupového systému, což je potvrzení PID_U o identitě uživatele U. Zde obvykle protokol OpenID Connect končí. My si však v dalším popíšeme celý běh protokolu, tj. vysvětlíme si jeho fungování i s nepovinným serverem IB (přístupový systém PS_{IB}). V uvedeném případě může server W získat na základě dokazovacího faktoru DF_{IB} aktiva přístupového systému PS_{IB}, jimiž jsou další údaje o identitě uživatele U (např. jeho občanské jméno a příjmení). Potřebný dokazovací faktor DF_{IB} vydává spolu s potvrzením PID_U autentizační brána AB.



Obrázek 3.11: Schéma protokolu OpenID Connect

Předpokladem fungování protokolu OpenID Connect je tzv. registrace uživatele i webového serveru u poskytovatele identity. V rámci registrace uživatele se sjedná uživatelova identita ID_U a sjedná se i technika jeho autentizace a příslušné autentizační faktory. V našem příkladu budeme předpokládat nejčastěji používanou techniku autentizace, kterou je autentizace heslem PSW_U. Potom dokazovací faktor $DF_U = PSW_U$ a ověřovací faktor $OF_U = \mathrm{HSF}(PSW_U)$. Kromě toho může uživatel poskytovateli identity svěřit i jiné své osobní údaje, jako je občanské jméno a příjmení, poštovní a e-mailová adresa apod. Tyto údaje jsou uloženy v serveru IB, který jsme nazvali informační brána. I v rámci registrace webového serveru W se sjedná jeho identita ID_W, technika jeho autentizace a příslušné autentizační faktory. Opět zde budeme předpokládat běžnou techniku autentizace, kterou je autentizace heslem typu BAA. Potom dokazovací faktor webového serveru $DF_W = PSW_W$ a ověřovací faktor $OF_W = \mathrm{HSF}(PSW_W)$.

V rámci protokolu OpenID Connect je veškerá komunikace se servery poskytovatele identity (tj. se servery AS, AB a IB) uskutečňována pomocí protokolu TLS. Certifikáty s veřejnými klíči těchto serverů jsou spolu s webovými adresami *ADR* serverů (tj. s adresami ADR_{AS}, ADR_{AB}, ADR_{IB}) předány protistranám v rámci jejich registrace. Prostřednictvím TLS spojení je zajištěna důvěrnost a autentičnost komunikace. Autentičnost stran je v případě serverů poskytovatele identity zajištěna prostřednictvím veřejných klíčů v certifikátech těchto serverů a v případě uživatele U a webového serveru W je autentičnost dané strany zajištěna jejím heslem. Tato hesla se sdělují serverům poskytovatele identity ve spoji zabezpečeném protokolem TLS. Výjimkou je spojení mezi serverem W a informační branou IB, kde se však server neautentizuje a prokazuje pouze znalost dokazovacího faktoru vydaného autentizační branou AB.

Nyní si vysvětlíme fungování protokolu. Na obrázku jeho průběhu vidíme, že některé přenosy jsou vyznačeny čárkovanými šipkami. Tyto přenosy standard nestanovuje a jsou ponechány na řešení zúčastněných stran. V prvním kroku uživatel U žádá od serveru W stránku *WS*. Protože však žadatel není doposud autentizován, tak server W zašle webovému prohlížeči uživatele U zprávu sestávající z bloků ID_W, PV_W a ADR_{AS}. Blok ID_W je identita serveru W, PV_W je žádost serveru W o právo zjistit a ověřit identitu uživatele U a případně získat k danému uživateli další informace (např. jeho e-mailovou adresu) a ADR_{AS} je adresou autentizačního serveru AS. Tato zpráva je zaslána spolu s příkazem tzv. přesměrování (obvykle HTTP kód 302). Na základě tohoto příkazu webový prohlížeč uživatele zruší spojení s webovým serverem W a naváže spojení se serverem na adrese ADR_{AS} (tj. se serverem AS). Tomu předá bloky ID_W a PV_W, takže se autentizační server dozví, že jej server W žádá o autentizaci k němu přesměrovaného uživatele a o práva získat další informace o tomto uživateli.

Autentizační server odešle webovému prohlížeči uživatele autentizační formulář. Uživatel v něm vyplní svůj identifikátor ID_U, své heslo PSW_U a tyto údaje autentizačnímu serveru odešle. Server ověří správnost hesla a v kladném případě uživateli zašle schvalovací formulář pro práva PV_W nárokovaná serverem W. Uživatel tato nárokovaná práva případně zredukuje na práva, která označíme PR_W a zašle je serveru AS. Dejme tomu, že uživatel svolil ke sdělení výsledku své autentizace a souhlasí s předáním svých vybraných osobních údajů serveru W. Server AS poté vygeneruje unikátní číslo (tzv. autorizační kód – „authorization code“), které má pro server W význam dokazovacího faktoru DF_{AB} vůči autentizační bráně AB. Server AS tento dokazovací faktor spolu s adresou serveru W (tj. adresou ADR_W) a s příkazem přesměrování odešle uživateli U. Webový prohlížeč uživatele pak zruší spojení s AS, naváže opět spojení s W a tomu předá dokazovací faktor DF_{AB}. Server AS mezitím zaslal autentizační bráně AB trojici (ID_U, PR_W, DF_{AB}). Pro tuto bránu (fakticky přístupový systém PS_{AB}), se jedná o příkaz od nadřízeného přístupového systému PS_{AS}. Obsahem tohoto příkazu je, že pokud se server W uvedený v bloku PR_W prokáže ověřovacím faktorem DF_{AB}, tak mu autentizační brána AB má identitu uživatele U potvrdit. A protože v bloku PR_W jsou uvedena práva k získání dalších osobních údajů, tak mu má k tomu vydat dokazovací faktor DF_{IB}.

Server W nyní naváže TLS spojení s autentizační branou AB a odešle jí svůj identifikátor ID_W, heslo PSW_W a dokazovací faktor DF_{AB}. Prostřednictvím identifikátoru ID_W se server W představí

a heslem PSW_W se zároveň autentizuje. Autentizační brána pak pomocí hodnoty DF_{AB} ve své paměti vyhledá trojici (ID_U, PR_W, DF_{AB}), kterou jí zaslal autentizační server. Z ní zjistí, jaká práva PR_W získal server W k informacím ohledně uživatele s identitou ID_U. Brána AB poté serveru W zašle získaná aktiva, kterými jsou jí podepsané potvrzení o identitě uživatele U (tzv. „ID Token" – na obrázku blok označený PID_U) a další autorizační kód (tzv. „Access Token"), který bude mít pro server W význam dokazovacího faktoru DF_{IB}. Současně autentizační brána AB zašle informační bráně IB trojici (ID_U, PR_W, DF_{IB}).

Server W nakonec naváže TLS spojení s informační branou IB a v něm ji zašle dokazovací faktor DF_{IB}. V tomto případě se server neautentizuje, protože DF_{IB} je tzv. dokazovacím faktorem na doručitele („Bearer Token"). Informační brána podle hodnoty DF_{IB} vyhledá ve své paměti trojici (ID_U, PR_W, DF_{IB}), kterou jí zaslala autentizační brána. Z ní zjistí, jaká práva PR_W k informacím ohledně uživatele s identitou ID_U získal odesílatel dokazovacího faktoru DF_{IB} (tj. server W). Příslušné osobní údaje pak brána IB odesílateli dokazovacího faktoru DF_{IB} předá (na obrázku blok $INFO_U$). Webový server má nyní potvrzenu identitu uživatele U. Jeho kontrolér na základě přístupového seznamu zjistí práva uživatele a podle nich následně reguluje jeho přístup ke stránkám serveru. Zároveň si webový server zapíše do své databáze získané informace $INFO_U$ (např. e-mailovou adresu uživatele), které může později využít například k zasílání nabídek svých dalších služeb.

3.4 Autorizační protokoly

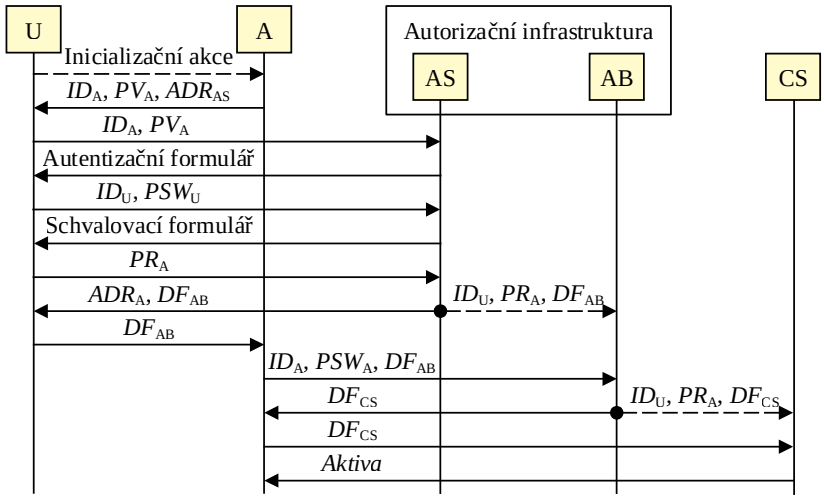
Autorizaci jsme definovali jako akt, během něhož autorita udělí žadatelovi Z přístupová práva PR_Z. Pro účely fungování přístupového systému s ním však autorita musí také sjednat jeho identitu ID_Z, dokazovací faktor DF_Z a ověřovací faktor OF_Z. Údaje ID_Z, PR_Z a OF_Z pak autorita uloží do přístupového systému. Autorizace, sjednání přístupových údajů a jejich následné vložení do přístupového systému probíhá buď v rámci osobního setkání autority a žadatele (např. při nástupu nového zaměstnance do firmy), nebo automatizovaně prostřednictvím příslušné aplikace (např. vyplněním formuláře a následným zaplacením přístupového poplatku ke zprávám na webovém serveru). Obě popsané možnosti mají společné to, že se jedná o přístup k aktivům jediné autority. V souvislosti s ukládáním dat tisíců uživatelů na sociální sítě však vznikl problém, jak tito uživatelé budou řídit přístup jiných uživatelů ke svým datům, které jsou však uloženy na cizích serverech. Odpovědí je protokol OAuth.

3.4.1 Protokol OAuth

Protokol OAuth [25] je autorizační protokol, jímž autorita může zájemcům o přístup udělovat přístupová práva ke svým aktivům, přičemž protokol se postará o automatické generování autentizačních faktorů a jejich distribuci zainteresovaným stranám. V případě protokolu OAuth je majitelem aktiv (a tedy autoritou) uživatel U, aktivy jsou jeho data na úložném serveru strany B

a žadatelem je aplikace A, která běží buď na serveru strany C, nebo na zařízení uživatele U (např. na smartfonu). K přenosu zpráv je použita kombinace protokolů HTTP a TLS.

Nejnovější verzí protokolu OAuth je jeho druhá verze, která definuje několik scénářů. My si zde popíšeme asi nejčastější scénář, kdy aplikace A běží na webovém serveru. Dejme tomu (viz obr. 3.12), že uživatel U má své fotografie (v tomto případě aktiva) uloženy na cílovém serveru CS, který je provozován stranou B. Předpokládejme nyní, že si uživatel chce nechat své fotografie vytisknout firmou C, která zakázky přijímá prostřednictvím aplikace A spuštěné na serveru této firmy. Poslední komponentou nutnou k fungování protokolu OAuth je autorizační infrastruktura tvořená autorizačním serverem AS a autorizační branou AB. Prostřednictvím protokolu OAuth může uživatel U udělit aplikaci A přístupová práva ke svým fotografiím na cílovém serveru CS. Aplikace potom může uživatelovy fotografie z cílového serveru zkopírovat a následně zajistit jejich tisk. Autorizační infrastruktura přitom zajišťuje autorizaci (tj. udělení práv) uživatelem, generuje potřebné autentizační faktory a tyto faktory bezpečně distribuuje zúčastněným stranám. Finální dokazovací faktor se předává aplikaci A a příslušný ověřovací faktor se dodá cílovému serveru CS.



Obrázek 3.12: Schéma protokolu OAuth

Protokol OAuth je základem již vysvětleného autentizačního protokolu OpenID Connect, a tak si registraci uživatele i aplikace znova popisovat nebudeme. Začneme od okamžiku, kdy si uživatel U pomocí aplikace A objednal tisk fotografií. Uživatel nyní na objednávkovém formuláři stiskl tlačítko Objednat (na obrázku označeno jako „Inicializační akce“) a tím se spustil protokol OAuth. Aplikace A nejprve zašle webovému prohlížeči uživatele U zprávu sestávající z bloků ID_A, PR_A a ADR_{AS}. Blok ID_A je identita aplikace A, blok PV_A jsou nárokovaná přístupová práva a ADR_{AS} je

adresou autorizačního serveru AS. Tato zpráva je opatřena příkazem přesměrování (HTTP kód 302), takže webový prohlížeč uživatele zruší spojení s webovou aplikací A a naváže spojení se serverem na adrese ADR_{AS} (tj. se serverem AS). Tomu předá bloky ID_A, PV_A, čímž se autorizační server dozví, že jej aplikace A žádá o to, aby jí právě připojený uživatel U přidělil práva PV_A.

Autorizační server poté odešle webovému prohlížeči uživatele autentizační formulář. Uživatel v něm vyplní svůj identifikátor ID_U, své heslo PSW_U a tyto údaje autorizačnímu serveru odešle. Ten ověří správnost hesla a v kladném případě uživateli zašle schvalovací formulář, v němž uživateli sdělí, jaká práva PV_A si aplikace nárokuje. Uživatel tato práva případně zredukuje do podoby práv PR_A a ta pak serveru AS zašle. Server AS nyní vygeneruje unikátní číslo (tzv. autorizační kód – „authorization code“), které má pro aplikaci A význam dokazovacího faktoru DF_{AB} vůči autorizační bráně AB. Server AS tento dokazovací faktor spolu s adresou aplikace A (tj. adresou ADR_A) a s příkazem přesměrování odešle uživateli U. Webový prohlížeč uživatele proto zruší spojení s AS, naváže opět spojení s aplikací A, které předá dokazovací faktor DF_{AB}. Server AS mezitím zaslal autorizační bráně AB trojici (ID_U, PR_A, DF_{AB}).

Aplikace A nyní naváže TLS spojení s autorizační branou AB a odešle jí svůj identifikátor ID_A, své heslo PSW_A a dokazovací faktor DF_{AB}. Heslem PSW_A se aplikace A vůči autorizační bráně autentizuje jako registrovaná aplikace s identifikátorem ID_A. Autorizační brána potom pomocí dokazovacího faktoru DF_{AB} vyhledá trojici (ID_U, PR_A, DF_{AB}), kterou jí zaslal autorizační server AS. Pokud ji ve své paměti nalezne, tak zašle aplikaci A autorizační kód (tzv. „Access Token“), který bude mít pro aplikaci A význam dokazovacího faktoru DF_{CS}. Současně autorizační brána AB zašle cílovému serveru CS trojici (ID_U, PR_A DF_{CS}).

Aplikace A posléze naváže TLS spojení s cílovým serverem CS a v něm mu předá dokazovací faktor DF_{CS}. Cílový server podle hodnoty DF_{CS} vyhledá ve své paměti trojici (ID_U, PR_A, DF_{CS}), kterou mu zaslala autorizační brána. Z ní zjistí, jaká práva PR_A k aktivům uživatele s identitou ID_U získal odesílatel dokazovacího faktoru DF_{CS} (tj. aplikace A). Cílový server pak aplikaci umožní v rozsahu těchto práv přístup k aktivům uživatele U (na obrázku data označená jako *Aktiva*).

Výše popsaný systém je prakticky třístupňový přístupový systém, v němž protokol OAuth zajišťuje autorizaci žadatelů a distribuci autentizačních faktorů. Žadatelem je aplikace A, autoritou je uživatel U a chráněnými aktivy jsou data uživatele. Hierarchicky nejvyšší přístupový systém je autorizační server AS, jehož aktivem je dokazovací faktor DF_{AB}. Pomocí tohoto kódu pak může žadatel (tj. aplikace A) získat od druhého přístupového systému (tj. od autorizační brány AB) další aktivum, kterým je dokazovací faktor DF_{CS}. A nakonec pomocí tohoto faktoru získá od třetího přístupového systému (tj. od serveru CS) cílová aktiva.

Za poznámku stojí, že autorizační brána AB vyhlíží v popsaném systému zdánlivě zbytečně, neboť dokazovací faktor DF_{CS} by aplikaci mohl vydávat přímo autorizační server AS. Důvodem existence autorizační brány je vyšší bezpečnost. Aplikace A totiž komunikuje s autorizačním serverem AS zprostředkovaně přes webový prohlížeč uživatele U, tj. bez toho, že by se aplikace vůči serveru

autentizovala. Aplikace proto nejprve dostane jen autorizační kód DF_{AB}. Následně pak musí navázat TLS spojení s autorizační bránou AB a v tomto spojení se autentizovat svým heslem. Po předložení autorizačního kódu DF_{AB} získá dokazovací faktor DF_{CS} a s jeho pomocí nakonec získá přístup k aktivům na cílovém serveru CS. Všimněme si rovněž, že třetí přístupový systém (tj. server CS) je bez autentizátoru. Přístupová práva jsou zde vázána na anonymního doručitele dokazovacího faktoru a nikoliv, tak jak je to běžné, na konkrétní identitu žadatele. Cílové servery tak nemusí řešit registraci a autentizaci nejrůznějších aplikací a spoléhají v tom na autorizační infrastrukturu.

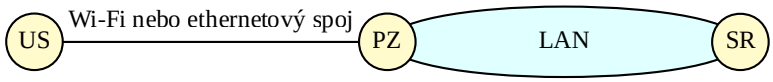
3.5 Přístupové protokoly

Přístupové protokoly zajišťují řízení přístupu v tzv. distribuovaných systémech, tj. v přístupových systémech, sestávajících z geograficky rozptýlených zařízení. Takovýmito systémy jsou typicky přístupové systémy s centralizovanou konfigurací. My se zde seznámíme s protokolem RADIUS a s obecným řešením systémů tzv. elektronické kontroly vstupu (EKV). V případě protokolu RADIUS je aktivem přístup ke službám sítě a v případě systémů EKV je aktivem možnost vstupu do místnosti či do budovy.

3.5.1 Protokol RADIUS

Protokol RADIUS („Remote Authentication Dial-In User Service“) [26] se typicky používá k řízení přístupu stanic uživatelů do počítačových sítí LAN. Rozhraní těchto sítí tvoří obvykle Wi-Fi přístupové body nebo přístupové přepínače. Přes tyto body, resp. přepínače (dále je budeme označovat jako přístupová zařízení PZ) se uživatelské stanice US připojují do sítě. Z hlediska architektury přístupových systémů jsou přístupová zařízení branami. Kontrolér i autentizátor jsou integrovány na jediném serveru, který pojmenujeme server RADIUS (SR).

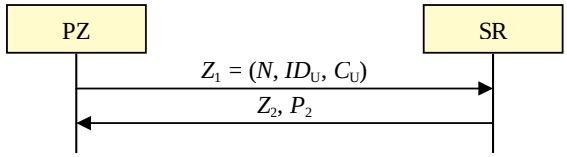
Protokol RADIUS je aplikační protokol, který definuje komunikaci mezi serverem SR a jím řízenými přístupovými zařízeními PZ (viz obr. 3.13).



Obrázek 3.13: Architektura prvků protokolu RADIUS

Uvedená komunikace se děje prostřednictvím počítačové sítě, přičemž k zabezpečení zpráv protokolu RADIUS slouží klíč K, který do jednotlivých zařízení ručně vkládá správce sítě. Komunikaci mezi přístupovým zařízením PZ a uživatelskou stanicí US protokol RADIUS neřeší.

V protokolu RADIUS je standardizována autentizace pomocí hesla. Předpokládá se přitom, že přístupové zařízení nějakým způsobem získá od uživatelské stanice US uživatelův identifikátor ID_U a uživatelovo heslo PSW_U. Z předchozího víme, že se tento přenos obvykle děje pomocí zpráv protokolu EAPoL. Přístupové zařízení poté vygeneruje náhodné číslo N a s jeho pomocí odvodí klíč K_{US} pro šifrování hesla PSW_U. Platí, že $K_{US} = \mathrm{ODF}(K, N)$, kde ODF je stanovená odvozovací funkce. Zařízení PZ pak serveru SR odešle zprávu $Z_1 = (N, ID_U, C_U)$, kde kryptogram $C_U = \mathrm{ENC}(PSW_U, K_{US})$, je uživatelovo heslo zašifrované klíčem K_{US} (viz obr. 3.14).



Obrázek 3.14: Schéma protokolu RADIUS

Server SR nejprve pomocí přijatého čísla N a výchozího klíče K odvodí hodnotu klíče $K_{US} = \mathrm{ODF}(K, N)$. Následně pak může z kryptogramu C_U dešifrovat uživatelovo heslo, neboť $\mathrm{DEC}(C_U, K_{US}) = PSW_U$. Pomocí své databáze pak ověří, zda uživatel s identifikátorem ID_U má skutečně přiděleno heslo PSW_U. Následně server SR zašle přístupovému zařízení zapečetěnou zprávu (Z_2, P_2), kde zpráva Z_2 obsahuje příkaz, zda má zařízení PZ danou stanici připojit, resp. nepřipojit do sítě. Protože příkaz má jen dvě možné hodnoty, tak se z bezpečnostních důvodů hodnota pečeti P_2 vypočítává z řetězce $(Z_2 \| N)$. Tím se zajišťuje unikátnost pečeti a také i její závislost na předchozí zprávě Z_1. Formálně tedy pro pečeť platí, že $P_2 = \mathrm{PCT}(Z_2 \| N, K)$. Přístupové zařízení nejprve ověří správnost pečeti a poté podle příkazu ve zprávě Z_2 umožní, resp. neumožní přístup stanice US do sítě.

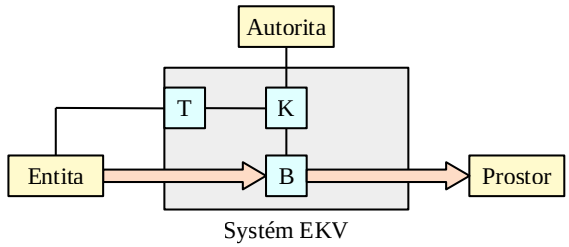
Kromě popsané výchozí autentizace umožňuje protokol RADIUS i jiné typy autentizace. Již jsme uvedli, že asi nejvíce se využívají autentizace pomocí protokolu EAP. V takovémto případě se zprávy protokolu EAP přenášejí v těle zpráv protokolu RADIUS jako přílohy těchto zpráv (tzv. „Attribute Value Pair" – AVP).

3.5.2 Systémy elektronické kontroly vstupu

Systémy elektronické kontroly vstupu (EKV) jsou přístupové systémy, jejichž nejčastějším aktivem je možnost vstoupit do chráněného prostoru, kterým je obvykle místnost, budova či areál [27]. Obecnou architekturu systému EKV vidíme na obr. 3.15.

Systém EKV sestává z kontroléru K, brány B a terminálu T. Kontrolér je řídícím prvkem systému a brána umožňuje žadatelům vstup do chráněného prostoru (typicky se jedná o elektricky ovládané dveře, brány či turnikety). Terminál T primárně zprostředkovává komunikaci žadatele

se systémem EKV a v moderních systémech zpravidla plní také roli autentizátoru A. Typicky se jedná o čtečky mikroprocesorových karet nebo o biometrické čtečky. Ve starších systémech plní roli autentizátoru kontrolér. Tehdy bývá terminálem klávesnice (autentizace heslem) či čtečka paměťových karet.



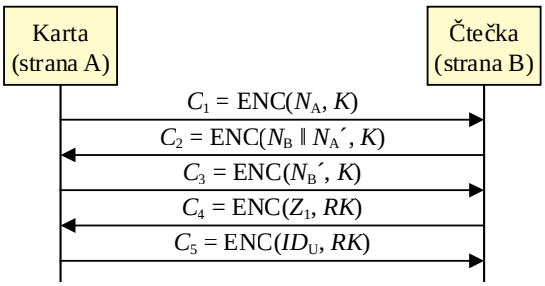
Obrázek 3.15: Architektura systému EKV

Pro nás jsou zajímavé systémy, kde terminálem je čtečka mikroprocesorových karet. Tento typ karet obsahuje mikropočítač, který umožňuje implementovat kryptografické techniky autentizace uživatelů. Každý uživatel U je v takovémto případě vybaven kartou, v jejíž paměti je uložen unikátní identifikátor uživatele ID_U a dokazovací faktor, kterým je tajný klíč K. Uvedený klíč obvykle sdílejí všechny karty i čtečky dané organizace. V praxi používané kartové autentizační protokoly se liší podle výrobců karet a nabízejí řadu variant. My si podle obr. 3.16 popíšeme jedno z často používaných řešení.

Uživatel zahajuje autentizační protokol tím, že přiloží svoji kartu (strana A) ke čtečce (strana B), jenž se nachází vedle dveří vedoucích do chráněné místnosti. Karta a čtečka spolu vytvoří bezdrátový přenosový kanál, kterým se přenášejí zprávy autentizačního protokolu.

Karta nejprve vygeneruje náhodné číslo N_A, které pomocí klíče K zašifruje do podoby kryptogramu $C_1 = \mathrm{ENC}(N_A, K)$. Tento kryptogram odešle čtečce. Čtečka kryptogram dešifruje, čímž získá číslo $N_A = \mathrm{DEC}(C_1, K)$. Z čísla N_A čtečka vytvoří nové číslo N_A' tak, že z něho odebere prvních 8 bitů a ty přemístí na konec zbylých bitů čísla N_A. Popsaná operace se nazývá rotace o 8 bitů doleva a my ji formálně vyjádříme jako ROL^8 („Rotation On Left“). Můžeme pak psát, že $N_A' = \mathrm{ROL}^8(N_A)$. Dále čtečka vygeneruje své vlastní náhodné číslo, které označíme N_B. Následně vytvoří kryptogram $C_3 = \mathrm{ENC}(N_B \| N_A', K)$, který poté odešle kartě. Ta kryptogram dešifruje, čímž získá řetězec $(N_B \| N_A') = \mathrm{DEC}(C_3, K)$. Karta tento řetězec rozpůlí na obě čísla N_B a N_A', přičemž zkontroluje, zda $\mathrm{ROL}^8(N_A)$ je rovno přijatému N_A'. V kladném případě protistrana (tj. čtečka) zná klíč K, čímž se čtečka vůči kartě autentizovala. Následně karta z dešifrovaného čísla N_B odvodí číslo $N_B' = \mathrm{ROL}^8(N_B)$, to zašifruje a čtečce je odešle jako kryptogram $C_3 = \mathrm{ENC}(N_B', K)$. Čtečka kryptogram dešifruje, čímž získá $N_B' = \mathrm{DEC}(C_3, K)$ a ověří si zda dešifrované N_B' je rovno hodnotě $\mathrm{ROL}^8(N_B)$. Je-li tomu tak, potom i karta prokázala čtečce znalost tajného klíče K, a tak se tímto

autentizovala. Náhodná čísla N_A a N_B byla přenášena šifrovaně, a proto mohou být použita jako semena, z nichž se odvodí klíč *RK* pro dané spojení (tzv. relační klíč). Formálně to zapíšeme, že $RK = \mathrm{ODF}(N_A \parallel N_B, T)$, kde ODF je určená odvozovací funkce a *T* je stanovený kontext.



Obrázek 3.16: Typický autentizační protokol v systému EKV

Po fázi autentizace stran a ustavení klíče začíná provozní fáze. Její popis si trochu zjednodušíme tím, že opomineme pečetění přenášených dat. Čtečka nejprve kartě zašle kryptogram $C_4 = \mathrm{ENC}(Z_1, RK)$, kde zpráva Z_1 je výzva k zaslání identifikátoru uživatele ID_U. Karta kryptogram dešifruje a na výzvu odpoví kryptogramem $C_5 = \mathrm{ENC}(ID_U, RK)$. Čtečka přijatý kryptogram dešifruje, čímž získá uživatelův identifikátor $ID_U = \mathrm{DEC}(C_5, RK)$. Tento identifikátor do karty vložila společně s tajným klíčem *K* autorita a dešifrovaná hodnota ID_U tedy nemůže být falešná. Čtečka identifikátor ID_U předá kontroléru K, který následně zjistí přístupová práva uživatele U. Pokud má daný uživatel právo v danou dobu vstoupit do příslušných dveří, tak mu kontrolér tyto dveře (bránu B) odemkne.

4 Platební systémy

4 Platební systémy

Platební systémy slouží k převodu peněz z účtu strany A na účet strany B. Nás pochopitelně budou zajímat elektronické systémy, v nichž se zaměříme na systémy využívající kryptografii. V této kapitole se blíže seznámíme s internetovým bankovnictvím, s protokolem 3D Secure, se sítí Bitcoin a nakonec s platebními kartami. Chceme-li uvedené systémy stručně charakterizovat, tak pomocí internetového bankovnictví může majitel účtu (dále klient) spravovat peníze na svém účtu, protokol 3D Secure klientům slouží zejména k platbám nákupů přes internet, síť Bitcoin se používá k platbám pomocí kryptoměny a platební karty se nejčastěji používají k bezhotovostním platbám nákupů v prodejnách.

Platební systém je prakticky specifickým druhem přístupového systému, přičemž aktivy jsou peníze na účtu klientů. Každý klient, z jehož účtu se převod provádí, (tzv. plátce) hraje roli autority, klient na jehož účet převáděné peníze směřují (tzv. příjemce) je fakticky žadatelem a platební systém (obvykle systém bank) hraje roli přístupového systému.

4.1 Internetové bankovnictví

Internetové bankovnictví je elektronický systém, který klientům umožňuje vzdálenou správu peněz na jejich účtu. Prostřednictvím uvedeného systému mohou klienti kontrolovat stav svého účtu a bance zadávat platební příkazy, jimiž se převádějí peníze z jejich účtu na jiný účet.

Internetové bankovnictví je založeno na komunikaci webového prohlížeče klienta s webovým serverem banky. Tato komunikace probíhá pomocí protokolu HTTP, který je kryptograficky zabezpečen nám již dobře známým protokolem TLS. Server banky se autentizuje pomocí svého certifikátu a klient se autentizuje ve vytvořeném TLS spojení obvykle pomocí hesla.

Pokud klient zadá bance příkaz provést převod peněz ze svého účtu na jiný (tzv. transakce), tak je serverem banky vyzván, aby zadanou transakci nějakým způsobem potvrdil. Často server banky zašle pomocí SMS („Short message service“) na telefon klienta potvrzovací číslo („Transaction Authentication Number“ – TAN). Jedná se o náhodné číslo, které pak klient zapíše do příslušné kolonky webového formuláře a toto číslo bance odešle. Ta přijatou hodnotu porovná s hodnotou, která byla uživateli zaslána a v případě shody je transakce provedena. Použitím potvrzovacího čísla se zvyšuje bezpečnost internetového bankovnictví, neboť útočník musí k úspěšnému útoku získat nejen heslo uživatele, ale i telefon uživatele. Z hlediska zabezpečení je v námi popsaném případě uživatel autentizován dvakrát – při přihlášení jde o autentizaci heslem a při potvrzování transakce jde o autentizaci hardwarem. Tato dvojí autentizace se často označuje jako tzv. dvoufaktorová autentizace.

Některé banky používají k potvrzování transakcí namísto telefonu zařízení, které nazveme potvrzovací kalkulátor. Tyto kalkulátory generují potvrzovací číslo buď na základě náhodné výzvy N,

kterou jim banka zašle, nebo na základě aktuálního času t. V ČR banky často používají potvrzovací kalkulátor RSA SecurID (obr. 4.1), v němž se využívá aktuální čas t.

Obrázek 4.1: Potvrzovací kalkulátor RSA SecurID (autor obrázku Mgr. Rudolf Burda)

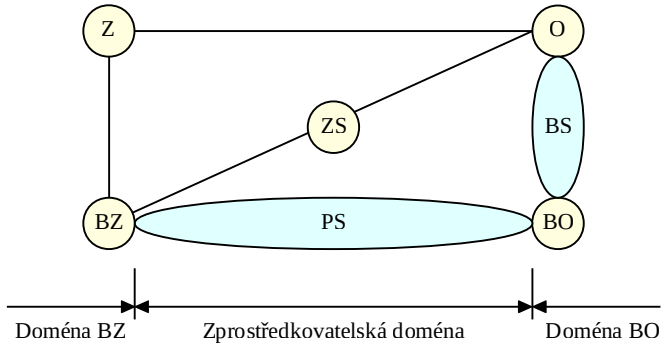
Princip je takový, že klientovi U je při založení účtu vydán potvrzovací kalkulátor s unikátním klíčem K_U. Tento kalkulátor i server banky mají synchronně jdoucí hodiny, takže aktuální čas na serveru je stejný jako na kalkulátoru. Kalkulátor s každou novou minutou vezme aktuální hodnotu času t, vypočítá k ní pečeť $P = \mathrm{PCT}(t, K_U)$ a vypočítanou hodnotu P zobrazí na svém displeji. Klient se v případě potvrzování nějaké transakce podívá na displej svého kalkulátoru a aktuální hodnotu pečeti odešle serveru banky. Server podle identifikátoru přihlášeného klienta zjistí klíč K_U a vypočítá kontrolní hodnotu pečeti $P' = \mathrm{PCT}(t, K_U)$. Pokud platí, že $P = P'$, tak je zaručeno, že protistrana disponuje potvrzovacím kalkulátorem klienta U a jedná se tedy o potvrzení zaslané klientem U.

4.2 Protokol 3D Secure

Protokol 3D Secure umožňuje zákazníkům s platební kartou platit internetové nákupy. Zde je však zapotřebí upozornit, že platební karty se v protokolu fyzicky vůbec nepoužívají a využívají se jen čísla na nich uvedená. Kryptografickým základem protokolu 3D Secure jsou spoje TLS. V současné době existuje protokol 3D Secure ve své druhé verzi [28], kterou si dále popíšeme. V uvedené verzi je definována varianta, kde zákazník platí pomocí vhodné aplikace (běžící například na smartfonu) a varianta, kde zákazník používá webový prohlížeč. My si vysvětlíme druhou z uvedených variant, a to jak pro scénář bez potvrzení zákazníkem, tak i pro scénář s potvrzením zákazníkem.

Základní prvky infrastruktury pro protokol 3D Secure vidíme na obr. 4.2. Tuto infrastrukturu lze rozdělit na tři domény. První doménu spravuje banka zákazníka BZ a kromě ní do této domény náleží uživatelské stanice zákazníků Z („3DS Client“). Druhou doménu má pod svojí správou banka obchodníka BO. Kromě ní je její součástí také internetový obchod obchodníka O („3DS Requestor“), přičemž obchodník komunikuje se svojí bankou prostřednictvím bankovní sítě BS. Do uvedené domény náleží i platební brána („3DS Server“), ale my si zde popis zjednodušíme a budeme popisovat variantu, kdy je platební brána integrována v serveru obchodníka O. Zbývající, třetí doména zprostředkovává interakce mezi prvky předchozích dvou domén. Tato

zprostředkovatelská doména obsahuje zprostředkovatelský server ZS a platební síť PS. Čáry na našem obrázku reprezentují spoje TLS. V TLS spojích směrem k uživatelské stanici Z se autentizují jen protistrany, tj. server BZ a server obchodníka O. K uvedené autentizaci oba servery používají běžné komerční certifikáty. V ostatních TLS spojích se provádí oboustranná autentizace pomocí veřejných klíčů podepsaných certifikační autoritou zprostředkovatelské domény. Předností doménového řešení je, že každá banka potřebuje dodržovat pouze standard 3D Secure. Ostatní bezpečnostní standardy (jako je například metoda autentizace zákazníků) může ve své doméně každá banka realizovat podle vlastní úvahy.



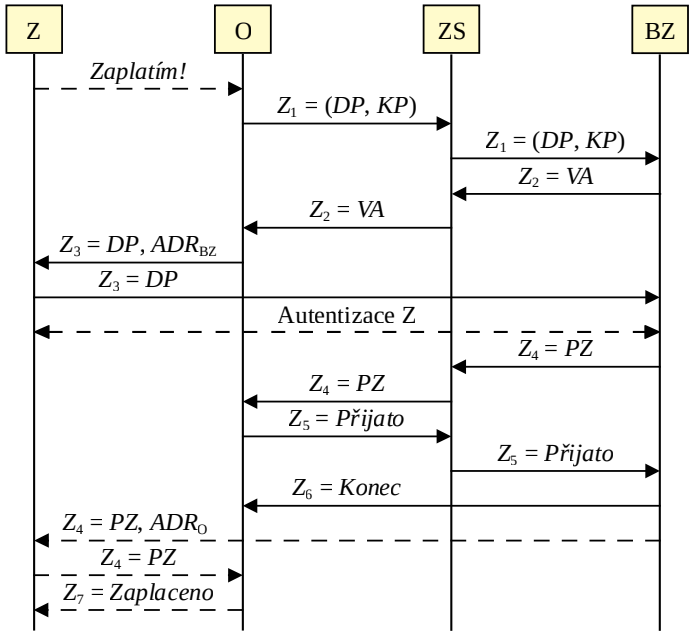
Obrázek 4.2: Infrastruktura pro protokol 3D Secure

Pokud se na architekturu protokolu 3D Secure podíváme z hlediska přístupových systémů, tak zákazník je zde autorita, aktivem jsou peníze na účtu zákazníka, žadatelem je internetový obchod a banka zákazníka hraje roli přístupového systému. Koncept protokolu je takový, že zákazník nejprve obchodníkovi přislíbí platbu. Prakticky se jedná o autorizaci, tj. o udělení práv k penězům zákazníka. Obchodník se s touto autorizací obrátí na banku zákazníka, tj. na přístupový systém. Banka tuto autorizaci buď přijme a vydá povolení k převodu (tzv. rychlá platba), nebo si nejprve vyžádá přímý příkaz od zákazníka a převod povolí teprve poté. V obou případech pak obchodník předá povolení k převodu své bance, která se postará o samotný převod.

Podrobněji si platbu pomocí protokolu 3D Secure vysvětlíme podle obr. 4.3. Kroky vyznačené souvislými šipkami jsou součástí standardu 3D Secure a čárkované šipky vyjadřují kroky, jejichž obsah si mohou zainteresované strany volit podle svého uvážení.

Zákazník Z si u internetového obchodníka O vybere zboží a objednávku potvrdí. Tím dojde k jeho přesměrování na integrovanou platební bránu, která mu zašle formulář, v němž vyplní své platební údaje. Těmito údaji jsou číslo jeho platební karty, platnost karty a ověřovací kód karty. Odesláním formuláře (na obrázku označeno „*Zaplatím!*“) se spustí běh protokolu 3D Secure. Obchodník O nejprve zformátuje zprávu $Z_1 = (DP, KP)$, kde DP jsou data platby a KP je kontext

platby. V datech platby jsou uvedeny údaje o zákazníkovi (zejména číslo karty), údaje o transakci (zejména cena a měna) a nakonec i údaje o obchodníkovi (zejména název obchodu a číslo účtu obchodníka). V kontextu platby *KP* jsou uváděny další údaje, které obchodník v průběhu objednávky o zákazníkovi zjistil (např. IP adresa a typ webového prohlížeče zákazníka). Zprávu Z_1 obchodník zašle zprostředkovatelskému serveru ZS. Ten podle čísla karty zákazníka zjistí banku zákazníka BZ a této bance zprávu Z_1 přepošle.



Obrázek 4.3: Schéma protokolu 3D Secure

V bance BZ se provede analýza rizik transakce. V rámci této analýzy se berou v úvahu data o platbě *DP*, kontext platby *KP* a historie plateb zákazníka (ta je uložena v databázi banky). Výsledkem této analýzy je buď zamítnutí transakce, nebo povolení transakce, přičemž toto povolení je buď nepodmíněné, nebo podmíněné. K zamítnutí transakce obvykle dochází, když zákazník nemá na svém účtu dostatečné peněžní prostředky. K nepodmíněnému povolení transakce dojde, pokud například zákazník již nějaké transakce s daným internetovým obchodem uskutečnil, nebo je cena transakce nízká. V případě podmíněného povolení transakce zákazníkova banka požaduje, aby jí zákazník danou transakci explicitně přikázal. Typicky se jedná o větší sumy, nebo zákazník používá neobvyklou IP adresu (je např. právě na dovolené v zahraničí). Výsledek analýzy, který označíme *VA*, banka BZ zašle zprostředkovatelskému serveru ZS v podobě zprávy $Z_2 = VA$ a ten ji předá obchodníkovi O.

Obchodník postupuje podle přijatého výsledku analýzy. Pokud banka zákazníka transakci zamítla, tak transakci zruší a informuje o tom zákazníka. V případě nepodmíněného povolení transakce je transakce schválena a obchod zákazníkovi zašle jeho zboží (tzv. rychlá platba – „Frictionless Flow"). V takovémto případě banka zákazníka ve své zprávě Z_2 zaslala i tzv. platební závazek PZ (ve standardu označováno „Authentication Value" – AV), což je pečeť této banky pro data o dané platbě, tj. PZ = PCT(DP, K), kde K je tajný klíč banky zákazníka. Na základě tohoto závazku si obchodník bude později nárokovat na bance BZ převod peněžních prostředků z účtu zákazníka na svůj účet. V obou výše uvedených případech protokol 3D Secure končí.

Pokud je výsledkem analýzy rizik podmíněné povolení transakce (náš příklad), tak protokol dále pokračuje. Obchodník O zašle webovému prohlížeči zákazníka zprávu Z_3 = DP a ADR_{BZ} spolu s příkazem přesměrování. Webový prohlížeč zruší TLS spojení k obchodníkovi, podle adresy ADR_{BZ} vybuduje TLS spojení k bance BZ a v něm jí předá data o platbě DP. Následně banka BZ zákazníkovi zašle formulář, s jehož pomocí se bude bance autentizovat. V tomto formuláři se již nacházejí vybraná data o platbě DP (obvykle cena, měna a název obchodu). Metoda autentizace závisí na bance zákazníka a obvykle se používají metody, které známe z předešlé kapitoly (tj. používá se buď bankou telefonicky zaslaný, nebo kalkulátorem vypočtený potvrzovací kód). Úspěšná autentizace se v tomto případě chápe jako explicitní zákazníkův příkaz k provedení zobrazené transakce.

V případě úspěšné autentizace pak zákazníkova banka BZ zašle přes zprostředkovatelský server obchodníkovi O zprávu Z_4 = PZ, kde PZ je již zmiňovaný platební závazek. Obchodník přijetí této zprávy potvrdí zprávou Z_5 = „*Přijato*" a běh protokolu ukončuje banka BZ zprávou Z_6 = „*Konec*". Banka zákazníka poté ještě může zákazníkovi zaslat zprávu Z_4 = PZ spolu s adresou obchodníka ADR_O a příkazem k přesměrování. Zákazníkův prohlížeč tak zruší spojení s bankou BZ, připojí se na server obchodníka a zprávu Z_4 = PZ mu předá. Server obchodníka si ověří, že tato zpráva mu již přišla od zprostředkovatelského serveru a zákazníkovi zašle zprávu Z_7 = „*Zaplaceno*". Zákazník tak má potvrzeno, že jeho platba proběhla v pořádku.

Jak již bylo uvedeno, tak internetový obchod po přijetí platebního závazku PZ považuje transakci za úspěšně ukončenou a zákazníkovi odešle jeho zboží. Za určité období (např. jednou za 24 hodin) obchodník shromáždí data o všech za tuto dobu provedených transakcích a předá je své bance BO, aby provedla vyrovnání s bankami všech zákazníků. Banka obchodníka s každou zákaznickou bankou komunikuje prostřednictvím neveřejné platební sítě PS (viz obr. 4.2). Od každé transakce jí zašle data o platbě DP a platební závazek PZ. Banka zákazníka tato data porovná se svými záznamy a pomocí svého tajného klíče K ověří, zda PCT(DP, K) = PZ. V kladném případě převede příslušnou sumu z účtu zákazníka na účet obchodníka.

4.3 Síť Bitcoin

Síť Bitcoin lze charakterizovat jako decentralizovanou platební síť, v níž jsou platidlem data (tzv. digitální měna). Slovo „decentralizovaná" vyjadřuje, že platební síť funguje autonomně na principu

vzájemně výhodné spolupráce navzájem nezávislých subjektů („peer-to-peer network"). V síti Bitcoin tedy neexistuje žádný centrální prvek, na němž by její fungování záviselo, což případným útočníkům komplikuje ovládnutí sítě. Základní platební jednotkou v síti je bitcoin (zkratka BTC), avšak v platebním styku se také používá satoshi, jehož hodnota je 10^{-8} BTC. Koncept bitcoinové sítě zformuloval v roce 2009 ve svém článku [29] anonym s přezdívkou Satoshi Nakamoto. Ten také dal na internet k dispozici potřebný software.

Síť Bitcoin tvoří navzájem propojené uzly, které podle jejich role rozdělíme na klienty, zpracovatele a servery. Klienti reprezentují jednotlivé účastníky plateb, tj. plátce a příjemce. Platby mezi klienty se uskutečňují pomocí software, které se nazývá peněženka („wallet"), přičemž vlastnictví bitcoinů klienti prokazují digitálním podpisem. Zpracovatelé (případně také „těžaři" z anglického „miners") zajišťují zejména převody bitcoinů mezi klienty a zhruba do roku 2140 zajišťují i produkci nových bitcoinů. Servery zajišťují připojování klientů a zpracovatelů do sítě Bitcoin. Rovněž tak kontrolují a zaznamenávají transakce a případně ostatním uzlům o těchto transakcích poskytují informace.

Síť Bitcoin funguje na základě aplikačního protokolu Bitcoin, přičemž zprávy tohoto protokolu jsou mezi uzly transportovány prostřednictvím protokolu TCP. K zapojení do sítě uzel potřebuje znát IP adresy několika páteřních serverů. Ty má buď uloženy ve své paměti, nebo si je zjistí pomocí služby DNS. Prostřednictvím páteřních uzlů se uzel dozví IP adresy dalších uzlů, s nimiž pak naváže spojení. Tímto způsobem je nad internetem vytvořena redundantní bitcoinová síť, což znamená, že každý uzel má vybudováno TCP spojení k více než jednomu dalšímu uzlu. Tato redundance jednak urychluje šíření zpráv v síti a současně zvyšuje bezpečnost transakcí. K oklamání klienta by případný útočník musel nejprve ovládnout všechny uzly, na něž je klient aktuálně připojen.

Zprávy protokolu Bitcoin se šíří lavinově. Uzel, když přijme zprávu, nejprve zkontroluje, zda již nepřijal její kopii od jiného uzlu. Pokud tomu tak není, tak zkontroluje formální i obsahovou správnost zprávy a poté ji odešle zbývajícím sousedům. Takovýmto způsobem se zprávy protokolu Bitcoin dostávají ke všem uzlům na této planetě se zpožděním řádově sekund.

Platby se uskutečňují pomocí tzv. platebního příkazu („transaction"). Platební příkaz, kterým klient A převádí bitcoiny na účet klienta B budeme značit PP_{AB}. Svůj příkaz plátce A podepíše a odešle do sítě Bitcoin. Zpracování platebních příkazů (tj. platební převody) realizují zpracovatelé, kteří tak fakticky plní úlohu banky. Zpracovatelé uskupí dosud neprovedené platební příkazy do platebních bloků *PB* („block") a tyto bloky se snaží napojit na tzv. platební historii („blockchain"). Platební historie jsou veškeré platební příkazy provedené za celou dobu existence sítě Bitcoin doplněné o kontrolní údaje, které umožňují kontrolu autentičnosti těchto příkazů. Platební historie je uložena na serverech ve veřejně dostupné databázi, takže kdokoliv může analýzou příkazů zjistit, který z klientů má kolik bitcoinů. Kompletní historii si často klienti či zpracovatelé při své instalaci kopírují ze serverů do svého lokálního paměťového úložiště, kde si ji pak samy průběžně aktualizují. K tomu, aby nebylo možné platební historii padělat, se používá technika tzv. kontrolního řetězce. Její princip je takový, že ke všeobecnému uznání příkazů v daném platebním bloku

PB je nutné heš tohoto bloku napojit na kontrolní řetězec. Toto napojení je však zcela záměrně konstruováno tak, aby bylo výpočetně náročné. K případnému ovládnutí sítě Bitcoin pak útočníci musí disponovat výpočetní kapacitou, která je větší než celková kapacita všech ostatních zpracovatelů. To se nejeví jako reálné.

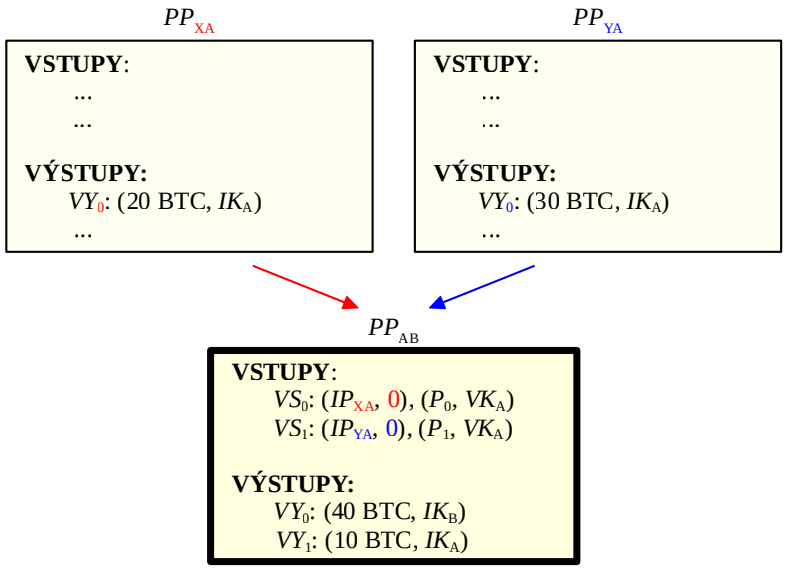
V síti Bitcoin se v současné době používá pět typů platebních převodů. My si zde popíšeme ten nejčastější, který se označuje zkratkou P2PKH („Pay-to-Public-Key-Hash"). V uvedeném případě si nový klient A nejprve vygeneruje podpisový kryptosystém se soukromým klíčem SK_A a veřejným klíčem VK_A. Z veřejného klíče si pak vytvoří bitcoinový účet s identifikátorem $IK_A = HSF_1(VK_A)$, kde HSF_1 je protokolem určená hešovací funkce. Využívá se zde bezkoliznost hešovacích funkcí, kdy různé vstupy hešovací funkce (v našem případě veřejné klíče) mají i různé výstupní heše (tj. identifikátory účtů). Použité řešení zajišťuje anonymitu klienta, protože jeho účet není spojen s jeho identitou, ale je vázán na vlastnictví kryptografického klíče. Klienti si mohou vygenerovat prakticky libovolné množství podpisových kryptosystémů, takže mohou být majiteli prakticky libovolného množství účtů. To se využívá k tomu, aby každá platba od ostatních plátců byla pokaždé uložena na jiný účet. Až do okamžiku případného společného použití těchto plateb potom nelze analýzou platební historie zjistit, že všechny tyto platby měly stejného příjemce. Pro zjednodušení si však popíšeme scénář, kdy klient pracuje s jedním účtem.

Již jsme si uvedli, že platební příkaz klienta A (nebo-li příkazce) ve prospěch klienta B (nebo-li příjemce) budeme značit PP_{AB}. Uvedený příkaz bude v platební historii vyhledáván pomocí svého indexu $IP_{AB} = HSF_2(PP_{AB})$, kde HSF_2 je protokolem určená hešovací funkce. Opět se zde využívá bezkoliznost hešovacích funkcí, kdy různé vstupy hešovací funkce (v tomto případě příkazy) mají i různé výstupní heše (tj. indexy).

Strukturu platebního příkazu (což je jedna ze zpráv protokolu Bitcoin) si popíšeme podle obr. 4.4. Každý platební příkaz prakticky sestává ze dvou částí. První část tvoří tzv. vstupy a druhou část tvoří tzv. výstupy. Jak vstupů, tak i výstupů může být v platebním příkazu více, a proto se navzájem odlišují indexem *j*, což je jejich pořadové číslo v seznamu vstupů, resp. výstupů daného příkazu. Počáteční hodnota indexu je přitom nula. Vstup v platebním příkazu definuje bitcoiny, jimiž se bude platit a obsahuje důkaz, že majitelem těchto bitcoinů je příkazce. Výstupy pak určují, na jaké účty budou bitcoiny ze vstupů převedeny. Vstupy, resp. výstupy budeme značit VS_j, resp. VY_j, kde *j* je jejich index.

Počet bitcoinů, které klient A vlastní, je dán součtem bitcoinů, které mu na jeho účet převedli jiní klienti a které doposud neutratil. Sumy z těchto platebních příkazů se však na účtu nekumulují a jsou stále vázány na příslušné příkazy. Dejme tomu, že klient A má na svém účtu dva platební příkazy, které ještě nepoužil k žádné platbě (tzv. disponibilní příkazy). První platební příkaz PP_{XA} je od klienta X a zní na 20 BTC a druhý je platební příkaz PP_{YA} od klienta Y a ten je na sumu 30 BTC. Disponibilní zůstatek klienta A je tak 20+30 = 50 BTC. Předpokládejme, že klient A chce nyní platebním příkazem uskutečnit platbu 40 BTC klientovi B. K uvedené platbě od klienta B obdržel identifikátor jeho účtu IK_B.

Platební příkaz pro výše uvedený příklad vidíme na obr. 4.4. V prvním vstupu VS_0 jsou dvě dvojice položek. V indexové dvojici (IP_{XA}, 0) je položka IP_{XA} indexem platebního příkazu PP_{XA} a položka 0 je indexem prvního výstupu v tomto příkazu. Uvedená dvojice jednoznačně odkazuje v platební historii na výstup VY_0 platebního příkazu PP_{XA} (na obrázku vlevo nahoře). Tento výstup uvádí, že klient X zaslal obnos 20 BTC na účet IK_A. Dvojici (P_0, VK_A) nazveme ověřovací, neboť ostatním uzlům sítě slouží k ověření, zda příkazce je majitelem předmětných 20 BTC a zda je daný příkaz autentický. Položka VK_A je veřejný klíč příkazce A a položka P_0 = PCT(Z_{AB}, SK_A) je příkazcův podpis zprávy Z_{AB}. Ta je fakticky příkazem PP_{AB}, z jehož vstupů jsou vyňaty ověřovací dvojice.



Obrázek 4.4: Příklad platebního příkazu v síti Bitcoin

Struktura vstupu VS_1 je obdobná. Indexová dvojice (IP_{YA}, 0) odkazuje v platební historii na první výstup platebního příkazu PP_{YA} (na obrázku vpravo nahoře), podle něhož klient Y zaslal na účet IK_A obnos 30 BTC. Ověřovací dvojicí (P_1, VK_A) je podpis P_1 zprávy Z_{AB} a veřejný klíč příkazce VK_A (obecně jsou však veřejné klíče různých vstupů různé). Pokud sečteme počty bitcoinů z obou vstupů, tak zjistíme, že v našem platebním příkazu jde o převod celkem 20+30 = 50 bitcoinů.

Nyní se podívejme na výstupy našeho platebního příkazu. První výstup VY_0 určuje, že ze vstupních 50 bitcoinů má jít klientovi B na jeho účet IK_B celkem 40 BTC. Výstup VY_1 pak stanovuje, že zbývajících 10 bitcoinů se má vrátit zpět na účet plátce, tj. na účet IK_A. Plátce může v platebním příkazu také stanovit odměnu pro zpracovatele, čímž zvýší šanci na rychlé provedení svého

příkazu. Délky platebních bloků jsou omezené a zpracovatelé do nich přednostně vybírají právě příkazy s odměnou. Odměna pro zpracovatele není definována pomocí výstupu, ale je dána jako rozdíl mezi součtem hodnot vstupů a součtem hodnot výstupů. Náš příklad je bez odměny zpracovateli, ale pokud by součet vstupů byl například 50,1 bitcoinu, tak odměna pro zpracovatele by činila 50,1–50 = 0,1 BTC.

Klient A nyní svůj platební příkaz PP_{AB} odešle ke zpracování do sítě. Příkaz se nejprve dostane k sousedním uzlům, které jej ihned zkontrolují. Ověřují přitom, zda uvedené vstupy vůbec existují, zda již nebyly utraceny a zda jimi může příkazce disponovat. Postup kontroly si budeme ilustrovat na vstupu VS_0 s indexovou dvojicí (IP_{XA}, 0) a ověřovací dvojicí (P_0, VK_A). Nejprve se pomocí indexové dvojice (IP_{XA}, 0) nalezne v platební historii odkazovaný výstup. Index IP_{XA} určuje příkaz PP_{XA} a index 0 označuje příslušný výstup v tomto příkazu. V našem příkladu se jedná o výstup VY_0 = (20 BTC, IK_A). Poté se v platební historii hledá, zda tento výstup není vstupem jiného příkazu. To by totiž znamenalo, že oněch 20 bitcoinů klient A již utratil.

Pokud není uváděný obnos utracen, tak se dále kontroluje, zda plátce A je skutečně majitelem těchto 20 BTC. K tomu slouží ověřovací dvojice (P_0, VK_A). Kontrolující uzel vypočítá identifikátor účtu IK_A', který odpovídá klíči VK_A. Podle pravidla pro odvozování identifikátorů účtů platí, že $IK_A' = HSF_1(VK_A)$. Tato hodnota je porovnána s hodnotou IK_A, kterou klient X uvedl ve svém výstupu. Pokud platí, že $IK_A' = IK_A$, tak VK_A odpovídá účtu z výstupu příkazu klienta X. Následně se pomocí VK_A zkontroluje příkazcův podpis zprávy Z_{AB}. Pokud je autentizační indikátor $W = VER(Z_{AB}, P_0, VK_A)$ roven hodnotě 1, tak je podpis platný. Příkazce tímto prokázal, že disponuje soukromým klíčem SK_A, ke němuž náleží veřejný klíč VK_A. A protože ke klíči VK_A náleží účet IK_A, tak tím příkazce prokázal, že je majitelem účtu, na nějž klient X převedl 20 BTC. Má tedy právo s těmito 20 bitcoiny nakládat.

Sousední bitcoinové uzly analogicky zkontrolují i druhý vstup VS_1 a následně ještě ověří, zda součet bitcoinů ve vstupech je větší či roven součtu bitcoinů ve výstupech. Pokud je vše v pořádku, tak je platební příkaz PP_{AB} vyhodnocen jako korektní a uzly si jej uloží do své paměti k ostatním platným, avšak doposud neprovedeným příkazům. Platné a doposud neprovedené příkazy, které si každý uzel ukládá do své paměti, nazveme rezervoár („pool“). Sousední uzly poté příkaz rozešlou svým sousedům, které postupují podobně, a tak se po několika sekundách náš příkaz objeví v rezervoáru všech uzlů v síti.

Popsali jsme si tvorbu a nezávislou kontrolu platebního příkazu a teď si vysvětlíme samotný převod bitcoinů mezi účty. Převody provádějí zpracovatelé tak, že ze svého rezervoáru vyberou ty, které hodlají zahrnout do svého bloku. Vybrané příkazy uspořádají do platebního bloku *PB*, přičemž reprezentantem každého bloku (tj. všech v něm zahrnutých platebních příkazů) je heš $b = HSF_3(PB)$, kde HSF_3 je určená hešovací funkce. Cílem každého zpracovatele je napojit svůj platební blok *PB* na platební historii, neboť za to získá odměnu. Aktuální platební historie je v tomto případě reprezentována tzv. kontrolním hešem k a napojení bloku na tuto historii spočívá v nalezení takové hodnoty s, pro kterou bude platit

$$h = \mathrm{HSF}_2(k \parallel b \parallel s) \leq Mx,$$

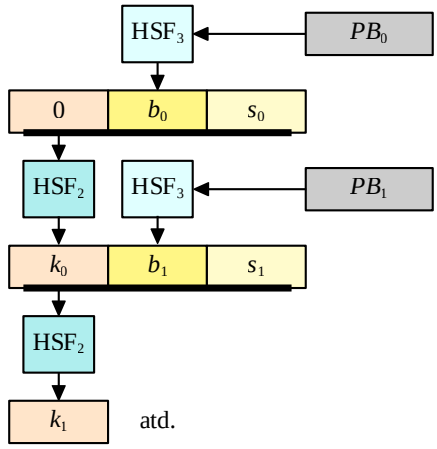
přičemž h je hodnota heše, HSF_2 je již zmiňovaná hešovací funkce a Mx je tzv. mezní hodnota. Hodnota Mx se stanoveným způsobem průběžně upravuje podle celkového výpočetního výkonu sítě tak, aby docházelo k napojení platebních bloků na platební historii v průměru jednou za 10 minut. Hodnotu s, pro níž platí výše uvedená nerovnost, nazveme spřáhlem, protože napojuje platební blok na platební historii podobně jako železniční spřáhlo napojuje vagón k vlaku.

Kvůli jednosměrnosti hešovacích funkcí není pro nalezení spřáhla znám žádný efektivní postup. Využívá se proto metoda postupného zkoušení možných hodnot a nalezení vhodné hodnoty s je tak věcí náhody. Ze všech zpracovatelů nalezne spřáhlo první ten, kdo bude mít více štěstí, přičemž samozřejmě větší výpočetní výkon jeho šanci na úspěch zvyšuje. Uvedené řešení zabraňuje centralizaci sítě – převody neprovádí jeden monopolní zpracovatel, ale všichni zpracovatelé se v těchto převodech víceméně náhodně střídají.

Zpracovatel, který první najde spřáhlo s, o této skutečnosti okamžitě informuje ostatní. Každý uzel si ověří, že pro zpracovatelem zveřejněné spřáhlo s a pro jeho platební blok PB platí, že heš $h = \mathrm{HSF}_2(k \parallel b \parallel s)$ je skutečně menší než Mx, přičemž $b = \mathrm{HSF}_3(PB)$ a k je stávající kontrolní heš platební historie. Všechny uzly následně zkontrolují správnost platebních příkazů v bloku PB, čímž se zabraňuje případným podvodům zpracovatele. V kladném případě si uzly tyto příkazy ze svých rezervoárů odstraní, platební historii o ně naopak napojením bloku PB rozšíří a novým kontrolním hešem k platební historie ustaví hodnotu h. Následně všichni zpracovatelé ihned skončí s dalšími pokusy o napojení svého stávajícího bloku na historii, z aktualizovaného rezervoáru příkazů vytvoří nový blok a ten se znovu pokoušejí napojit na právě aktualizovanou platební historii.

Kontrolní heše k, heše napojených platebních bloků b a příslušná spřáhla s se v čase mění. Pokud je budeme indexovat podle jejich pořadí v čase, tak vývoj platební historie můžeme ilustrovat na obr. 4.5.

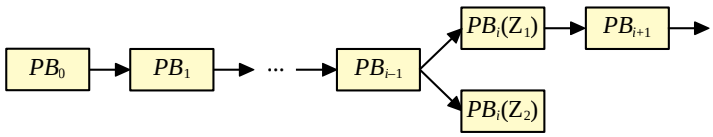
Autor sítě Bitcoin nejprve vytvořil výchozí platební blok PB_0, jenž obsahoval jediný příkaz, a to příkaz k emisi prvních 50 bitcoinů (viz dále). Následně vypočítal heš $b_0 = \mathrm{HSF}_3(PB_0)$, ten zřetězil s blokem ze samých nul (žádný kontrolní heš v té době doposud neexistoval) a k uvedenému řetězci nalezl spřáhlo s_0 tak, aby výsledný heš $k_0 = \mathrm{HSF}_2(0 \parallel b_0 \parallel s_0)$ byl menší než stanovená hodnota Mx. Heš k_0 se stal kontrolním hešem platební historie dané emisním příkazem v bloku PB_0. Následující platební příkazy pak byly uspořádány do bloku PB_1 a reprezentovány hešem $b_1 = \mathrm{HSF}_3(PB_1)$. K hodnotám k_0 a b_1 bylo poté nalezeno spřáhlo s_1 tak, aby výsledný heš $k_1 = \mathrm{HSF}_2(k_0 \parallel b_1 \parallel s_1)$ byl opět menší než stanovená hodnota Mx. Heš k_1 se tak stal kontrolním hešem platební historie dané platebními příkazy v blocích PB_0 i PB_1. Analogicky se postupuje dále až do současnosti. Každý nový kontrolní heš k_i umožňuje ověřit autentičnost platebních bloků PB_0 až PB_i.



Obrázek 4.5: Kontrolní řetězec v síti Bitcoin

Řetězec trojic $(0 \parallel b_0 \parallel s_0), (k_0 \parallel b_1 \parallel s_1), (k_1 \parallel b_2 \parallel s_2)$ atd. nazveme kontrolní řetězec, protože uzlům sítě umožňuje kontrolovat autentičnost platební historie a útočníkům tak znemožňuje podvádět (viz dále).

Podle pravidel sítě Bitcoinu musí zpracovatelé napojovat své platební bloky vždy na nejdelší větev platební historie. Může se totiž stát, že například zpracovatelé Z_1 a Z_2 své bloky napojí na platební historii prakticky současně. Vzhledem k lavinovému šíření zpráv bude ta část zpracovatelů, kteří jsou z hlediska síťové topologie blíže zpracovateli Z_1, akceptovat platební historii s jeho platebním blokem $PB_i(Z_1)$ a zbytek se bude snažit o navázání svého bloku na platební historii s blokem zpracovatele Z_2, tj. na blok $PB_i(Z_2)$. Dojde tím k rozvětvení platební historie (viz obr. 4.6). To se může opakovat i vícekrát. Nakonec však některý ze zpracovatelů bude v jedné z větví platební historie o dostatečný počet sekund rychlejší než zpracovatelé v ostatních větvích. Zpráva o jeho úspěchu všechny zpracovatele v jiných větvích zastaví, ti se okamžitě napojí na jeho větev a vývoj v ostatních paralelních větvích tím automaticky skončí. Platební příkazy, které se nacházejí jen v opuštěných větvích (tj. nejsou také zároveň v hlavní větvi) nejsou považovány za provedené a v nich převáděné bitcoiny nejsou jejich adresátům přiznány. Takovéto příkazy se automaticky vracejí zpět do rezervoárů.



Obrázek 4.6: Větvení platební historie

Zdrojem bitcoinů je proces zpracování platebních příkazů. Zpracovatel na první místo svého platebního bloku umisťuje tzv. emisní platební příkaz, který nemá žádné vstupy a má jen jeden výstup, a to x bitcoinů na účet zpracovatele. Pokud se zpracovateli podaří svůj blok na platební historii napojit, tak těchto x bitcoinů získá. Platí přitom, že hodnota $x = e+p$, kde e je počet nově emitovaných bitcoinů a p je součet odměn pro zpracovatele, které jsou stanoveny v platebních příkazech bloku. Nově emitované bitcoiny se dostávají k ostatním uživatelům sítě přes útraty zpracovatelů a aby se zamezilo inflaci, tak je hodnota e regulována. Regulační pravidlo stanovuje, že se vždy po zpracování 210.000 bloků hodnota e půlí. Při vzniku sítě byl počet emitovaných bitcoinů stanoven na hodnotu e = 50 BTC a protože už došlo ke dvěma půlením, tak hodnota e nyní činí 12,5 BTC. Regulační pravidlo vede k tomu, že počet bitcoinů nakonec bude nejvýše 21 miliónů a všechny se do oběhu dostanou kolem roku 2140. Motivací pro zpracovatele pak již budou jen platební příkazy, v nichž plátce přikazuje, aby část bitcoinů ze vstupů platebního příkazu šla na účet úspěšného zpracovatele.

Nyní si probereme možné útoky v síti Bitcoin. V případě klientů přicházejí v úvahu pouze pokusy o dvojí utracení bitcoinů v tzv. platbách bez potvrzení. V případě plateb bez potvrzení obchodníci akceptují již pouhé vydání platebního příkazu a nečekají několik minut na potvrzení převodu, tj. na zahrnutí daného platebního příkazu do platební historie. S platbami bez potvrzení se setkáváme v obchodech, kde se prodává zboží, jehož cena je nízká a doba pro jeho předání zákazníkovi je krátká. Příkladem může být prodej šálku kávy v kavárně. Platbu bez potvrzení by podvodník mohl využít například tím způsobem, že by vydal dva platební příkazy. Příkazem PP_1 by nařídil platbu obchodníkovi a příkazem PP_2 by ze stejných vstupů nařídil platbu sám sobě, přičemž by do tohoto příkazu zahrnul i odměnu zpracovateli. Tím by mohl dosáhnout toho, že zpracovatelé by jeho příkaz PP_2 zpracovali přednostně a příkaz pro obchodníka PP_1 by se potom při zpracování následujícího bloku dostal do rozporu s platební historií (příslušné vstupy již byly utraceny příkazem PP_2 v předchozím bloku). Podvodník by tímto způsobem získal obchodníkovo zboží v ceně odměny zpracovateli a obchodník by nedostal nic. K eliminaci popsaného útoku se používá takové opatření, kdy uzly do svého rezervoáru nepustí příkaz, který má v sobě stejný vstup jako některý z příkazů, který již v rezervoáru je. Uzly tak přijmou příkaz PP_1 (ten pak také obchodník vidí v rezervoáru svého uzlu), ale příkaz PP_2 již všechny uzly odmítnou. Mimochodem, právě v tomto většinovém dodržování pravidel sítě Bitcoin spočívá její bezpečnost. Pokud jsou útočníci v menšině, tak své podvody nedokáží v nějakém významném rozsahu prosadit. Pokud by klient poslal platební příkazy v opačném pořadí, tak by v rezervoáru obchodníkova uzlu byl příkaz PP_2 a příkaz PP_1 by se tam nedostal. Obchodník by jej tedy neviděl a zboží by klientovi nevydal.

Širší možnosti útoků mají zpracovatelé. Jedním z možných útoků je modifikace platební historie, čemuž však zabraňuje kontrolní řetězec. Již jsme uvedli, že řetězec trojic $(0 \| b_0 \| s_0)$, $(k_0 \| b_1 \| s_1)$, $(k_1 \| b_2 \| s_2)$ atd. na obr. 4.5 je kontrolní řetězec, který uzlům sítě umožňuje kontrolovat autentičnost platební historie. V této souvislosti je ještě vhodné připomenout, že případný útočník se může pokoušet jen o změnu jím vydaných příkazů, neboť soukromé podpisové klíče jiných klientů nezná. Předpokládejme tedy například, že útočník chce pozměnit nějaký svůj platební příkaz z bloku PB_1. Případná změna útočníkova platebního příkazu v bloku PB_1 však povede k tomu, že vznikne nový

blok PB_1' s novým hešem b_1'. Útočník pak musí nalézt novou hodnotu spřáhla s_1' tak, aby výsledný kontrolní heš k_1' byl roven původní hodnotě k_1. V důsledku bezkoliznosti hešovacích funkcí je však takovýto útok prakticky nemožný.

Druhou a reálnější možností útoku zpracovatele je opět dvojí útrata. U tohoto útoku se využívá skutečnost, že doba hledání spřáhla je sice v průměru 10 minut, ale v praxi je tato náhodná veličina velmi variabilní a může se pohybovat v rozsahu od sekund až po téměř hodinu. Podobně jako v případě klienta nečestný zpracovatel (označme jej zpracovatel Z_1) vytvoří dva platební příkazy se stejnými vstupy. Výstup příkazu PP_1 je adresován na obchodníka a výstup příkazu PP_2 jde zpět na účet zpracovatele. Předpokládejme nyní, že se právě připravuje zpracování i-tého bloku PB_i platební historie. Příkaz PP_1 rozešle zpracovatel Z_1 sousedním uzlům, takže ostatní zpracovatelé Z_x tento příkaz zahrnou do svých platebních bloků $PB_i(Z_x)$ a tyto se budou snažit napojit na platební historii. Příkaz PP_2 zpracovatel Z_1 vloží do svého bloku $PB_i(Z_1)$, čímž začne skrytý závod, během něhož zpracovatel Z_1 bude usilovat o prosazení platební historie s příkazem PP_2. Dejme tomu, že zbytku sítě se již podařilo napojit na historii celkem X bloků a i když zpracovatel Z_1 napojil také X bloků, tak se to pokaždé stalo o něco později. Pokud se mu však nyní podaří napojit další blok dříve než zbytku sítě, tak svůj závod vyhrál. Jeho větev bude v této chvíli o jeden blok delší a zveřejněním svých bloků docílí jejich zařazení do platební historie. Obchodník přislíbené bitcoiny nedostane.

Pravděpodobnost úspěchu výše uvedeného útoku se s rostoucím X exponenciálně snižuje. Obranou je proto opatření, kdy obchodník vyčkává s poskytnutím zboží nebo své služby do té doby, než bude platební příkaz PP_1 tzv. mnohonásobně potvrzen. X-násobným potvrzením se přitom rozumí, že za platebním blokem, v němž se příkaz PP_1 nachází (tzv. první potvrzení), je k platební historii ještě napojeno dalších (X–1) platebních bloků. V praxi se často považuje za dostatečnou hodnotu $X = 6$, což odpovídá zhruba hodinovému vyčkání obchodníka poté, co od plátce obdržel platební příkaz ve svůj prospěch. Pro představu předpokládejme, že nečestný zpracovatel Z disponuje celkem 10% výpočetního výkonu sítě a ostatní zpracovatelé mají 90% výkonu. Pravděpodobnost p události, že nečestný zpracovatel s tímto výkonem bude při hodnotě $X = 6$ úspěšný, je podle matematického modelu v [29] rovna hodnotě $p \approx 2{,}5 \cdot 10^{-4}$. Takovéto riziko je většinově považováno za přijatelné, přičemž dalším zvyšováním hodnoty X lze tuto pravděpodobnost (a tedy i riziko pro obchodníka) podle potřeby dále snižovat.

4.4 Platební karty

Platební karta, u níž se využívá kryptografie, je plastiková kartička (viz obr. 4.7) se zalisovaným jednočipovým počítačem. K zajištění komunikace s protistranou a ke svému napájení disponuje tento počítač buď kontaktním rozhraním, bezdrátovým rozhraním, nebo má tato rozhraní obě.

Pokud karta disponuje kontaktním rozhraním, tak se přenos energie pro čip a oboustranný přenos dat uskutečňuje pomocí vodičů, které jsou uspořádány do podoby kontaktního pole (na

levém obrázku v modrém rámu). V případě bezdrátového rozhraní se přenos energie pro čip a oboustranný přenos dat zajišťuje transformátorovou vazbou. V tomto případě je podél obvodu karty zalisován tenký drátek, který tvoří sekundární cívku. Odpovídající primární cívka se nachází ve čtečce karty. Přiblížením karty ke čtečce obě tyto cívky vytvoří transformátor, jímž se přenáší jak energie ze čtečky do čipu karty, tak i oboustranně data mezi kartou a čtečkou. Bezdrátový typ rozhraní lze na kartě identifikovat podle ikony, která je na obrázku zarámována ve žlutém čtverci. Na líci karty si ještě povšimněme červeně zarámované šestnáctimístné unikátní číslo, které nazveme identifikátor karty ID_K. Z tohoto identifikátoru lze určit banku, která je vlastníkem dané karty a v rámci této banky pak lze jednoznačně určit i držitele karty (nebo-li klienta). Na rubu karty (obrázek vpravo) je pro nás zajímavý magnetický proužek (v zeleném rámečku), na němž jsou zapsány identifikační a další údaje karty (zejména ID_K).

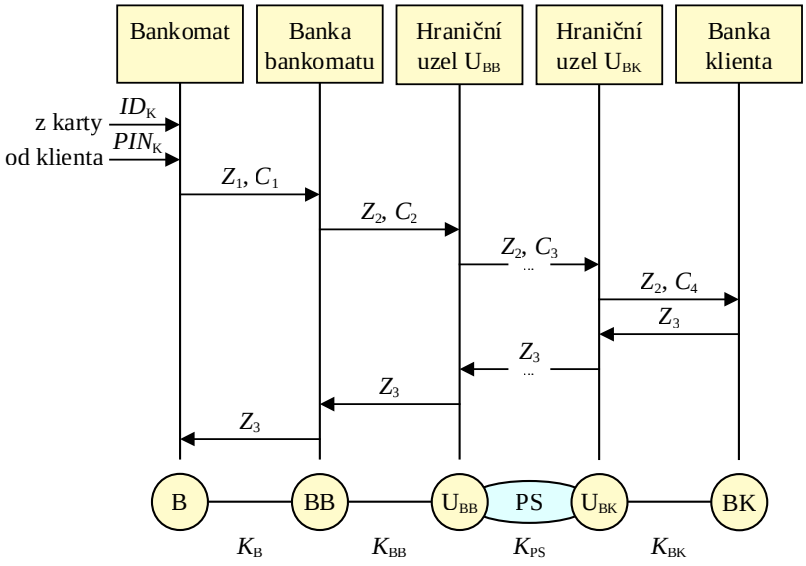
Obrázek 4.7: Platební karta s jednočipovým počítačem (autor obrázku Mgr. Rudolf Burda)

Platební karty se především používají k výběru hotovosti z bankomatů (obr. 4.8 vlevo) a k bezhotovostním platbám v prodejnách prostřednictvím platebních terminálů (obr. vpravo).

Obrázek 4.8: Bankomat (vlevo) a platební terminál (vpravo) (autor obrázku Mgr. Rudolf Burda)

Kryptografické zabezpečení platebních transakcí definuje standard [30]. My si zde popíšeme tři scénáře použití platebních karet. Prvním bude výběr hotovosti z bankomatu a druhým, resp. třetím scénářem bude platba v prodejně malou, resp. velkou částkou.

Nejprve si vysvětlíme výběr hotovosti z bankomatu (viz obr. 4.9). Na této transakci se podle spodní části obrázku podílí bankomat B, který je připojen ke své bance BB. Tato banka je přes hraniční uzel U_{BB} připojena do neveřejné platební sítě PS, do níž je přes hraniční uzel U_{BK} připojena i banka klienta BK. Ve všech spojích se data vesměs přenášejí nešifrovaně. Případná důvěrná data se utajují technikami symetrické kryptografie v linkové vrstvě. K tomu je mezi bankomatem B a jeho bankou BB ustaven klíč K_B, mezi bankou BB a jejím vstupním uzlem U_{BB} do platební sítě je použit klíč K_{BB}, uzly platební sítě spolu sdílejí společný klíč K_{PS} a banka klienta BK má vůči svému přípojnému uzlu U_{BK} sjednán klíč K_{BK}.



Obrázek 4.9: Výběr hotovosti z bankomatu

V případě výběru hotovosti se nevyužívá čip v kartě, ale jen její magnetický proužek. Důvodem pro použití této starší technologie je, aby si klienti mohli vyzvedávat hotovost prakticky kdekoliv, tj. i ve státech, kde je k dispozici jen technicky starší bankovní infrastruktura. Pokud klient zasune svoji kartu do čtecího otvoru bankomatu B, tak se z magnetického pásku karty načte identifikátor karty ID_K. Následně je klient vyzván, aby zadal PIN_K, kterým bude dokazovat svoji identitu vůči své bance BK. Bankomat poté nechá klienta zvolit částku CT vyžadované hotovosti a následně vytvoří zprávu $Z_1 = ID_K \parallel ID_B \parallel CT$, kde ID_K je identifikátor karty, ID_B je identifikátor bankomatu a CT je

požadovaná částka. Dále bankomat vypočítá kryptogram C_1 v němž zašifruje PIN_K klienta klíčem K_B, tj. $C_1 = \mathrm{ENC}(PIN_K, K_B)$. Dvojici Z_1 a C_1 pak bankomat odešle své bance. V bance BB se kryptogram C_1 dešifruje a získá se tak PIN_K klienta. Podle identifikátoru ID_K se zjistí, že se jedná o klienta jiné banky, a tak se PIN_K opět zašifruje, nyní do podoby kryptogramu $C_2 = \mathrm{ENC}(PIN_K, K_{BB})$. Následně se ve zprávě Z_1 nahradí identifikátor bankomatu identifikátorem banky ID_{BB}. Vznikne tak žádost o povolení transakce $Z_2 = ID_K \parallel ID_{BB} \parallel CT$. Dvojice Z_2 a C_2 je následně odeslána hraničnímu uzlu U_{BB} platební sítě. Ten pomocí jemu známých klíčů K_{BB} a K_{PS} přešifruje kryptogram z podoby C_2 do formy $C_3 = \mathrm{ENC}(PIN_K, K_{PS})$ a dvojici Z_2 a C_3 předá do platební sítě. V platební síti je dvojice Z_2 a C_3 mezi uzly směrována na základě identifikátoru ID_K, až je nakonec doručena do hraničního uzlu U_{BK}. Ten pomocí jemu známých klíčů K_{PS} a K_{BK} kryptogram C_3 přešifruje do podoby kryptogramu $C_4 = \mathrm{ENC}(PIN_K, K_{BK})$. Dvojici Z_2 a C_4 pak předá bance klienta BK.

Banka BK z doručené zprávy $Z_2 = ID_K \parallel ID_{BB} \parallel CT$ zjistí, že její klient s identifikátorem ID_K si chce v bankomatu banky BB vyzvednout hotovost ve výši CT. Dešifrováním kryptogramu C_4 zjistí PIN_K, který klient v bankomatu zadal a ve své databázi si ověří jeho správnost. Správnost pinu se zpravidla ověřuje vhodnou pečetící funkcí. Banka má ve své databázi ověřovací pečeť pinů svých klientů $PO_K = \mathrm{PCT}(PIN_K, HK_{BK})$, kde HK_{BK} je hlavní klíč banky. Pro dešifrovaný pin PIN_K banka vypočítá kontrolní pečeť $P_K = \mathrm{PCT}(PIN_K, HK_{BK})$. Pokud $P_K = PO_K$, tak je správnost pinu potvrzena. Dále si banka ověří, zda se na účtu daného klienta požadovaná hotovost nachází. Pokud tomu tak je, tak si zprávu Z_2 uloží, částku ve výši CT na účtu klienta zablokuje a stejnou cestou, ale opačným směrem bance BB odešle zprávu Z_3, že transakce je akceptována. Banka BB tuto zprávu předá bankomatu a ten požadovanou částku klientovi vydá. Mezi bankami periodicky (např. jednou za 24 hodin) probíhá tzv. dávkové vyrovnání. Banka BB zašle bance BK všechny zprávy Z_2 ze vzájemných transakcí dané periody. Banka BK si každou zaslanou zprávu porovná s uloženou kopií a v případě jejich shody dojde k převodu částky CT z účtu ID_K na účet banky BB.

Z důvodu bezpečnosti přenášených klientských pinů se jejich přešifrování provádí v tzv. modulech SAM („Security Application Module“). Tato zařízení jsou fyzicky střežena a technicky zabezpečena vůči různým invazivním i neinvazivním metodám zjišťování dat, které se uvnitř těchto modulů zpracovávají. Případný útočník, včetně správce modulu, tak není schopen zjistit, jaké piny se v paměti modulu SAM zpracovávají. Klíče do modulů SAM vkládají správci klíčů pro jednotlivé spoje.

V dalších scénářích se ke kryptografickému zabezpečení transakcí již využívá mikropočítač karty a k tomu si nejprve vysvětlíme správu klíčů. Mikropočítače v platební kartě jsou dostatečně výkonné, a tak kromě klíčů pro symetrickou kryptografii řeší tato správa rovněž klíče pro asymetrickou kryptografii. Pro bezpečné ustavení veřejných klíčů slouží dvoustupňová certifikační hierarchie. Hierarchicky nejvyšší je certifikační autorita, kterou nazveme globální autoritou GA. Tato autorita má svůj podpisový klíč SK_{GA} a odpovídající verifikační klíč VK_{GA}. Klíč VK_{GA} je ve formě kořenového certifikátu $CRT_{GA}(VK_{GA})$ již z výroby bezpečně uložen ve všech platebních terminálech, takže každý platební terminál má tento klíč k dispozici. Hierarchicky nižší jsou certifikační autority jednotlivých bank. Certifikační autorita každé banky BK má vygenerován svůj

podpisový klíč SK_{BK} a verifikační klíč VK_{BK}. Pro tento verifikační klíč VK_{BK} jí globální autorita GA vydá na požádání certifikát $CRT_{GA}(VK_{BK})$. Banka pak pro každou klientskou kartu K zajistí unikátní asymetrický kryptosystém se soukromým klíčem SK_K a veřejným klíčem VK_K. Tento asymetrický kryptosystém umožňuje jak podepisování, tak i šifrování. Zmíněnou dvojici klíčů si vytvoří sám mikropočítač karty, nebo ji do něj importuje banka. Certifikační autorita banky pak pro každou kartu ještě vytvoří certifikát $CRT_{BK}(VK_K)$ veřejného klíče karty a ten spolu s certifikátem $CRT_{GA}(VK_{BK})$ uloží do paměti karty.

Kromě asymetrické kryptografie se v platebním systému využívá i symetrická kryptografie. Každá banka BK má svůj hlavní klíč HK_{BK}. Z něho se pak pro každou kartu odvozuje hlavní klíč HK_K karty tak, že $HK_K = ODF_1(HK_{BK}, ID_K)$, kde ODF_1 je určená odvozovací funkce a kontext ID_K je identifikátor karty. Pro každou jednotlivou transakci se nakonec generuje relační klíč RK_K karty tak, že $RK_K = ODF_2(HK_K, NT_K)$, kde ODF_2 je určená odvozovací funkce a kontext NT_K je počet doposud uskutečněných transakcí dané karty K. Tím se dosahuje toho, že klíč pro každou transakci bude pokaždé jiný.

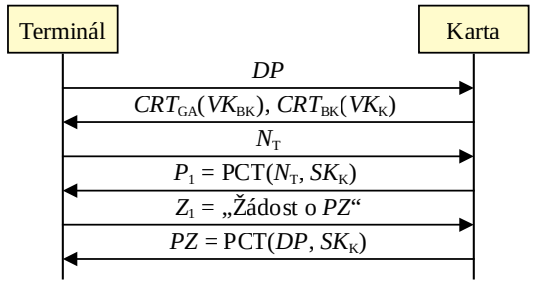
Pokud si to shrneme, tak se v kartě nacházejí klíče SK_K a VK_K kryptosystému, který kartě slouží jak k podepisování, tak i k šifrování. V paměti karty je rovněž uložen certifikát bankovní autority $CRT_{BK}(VK_K)$ veřejného klíče karty, dále certifikát globální autority $CRT_{GA}(VK_{BK})$ veřejného klíče banky a nakonec i hlavní klíč HK_K karty. Kromě těchto parametrů je v paměti karty uložen ještě ověřovací faktor identity klienta, kterým je číselné heslo (nebo-li pin) PIN_K.

Platební protokol mezi kartou a platebním terminálem je dosti komplikovaný. Po navázání spojení mezi kartou K a terminálem T se však v zásadě uskuteční následující fáze.

1. Zahájení transakce. Karta a terminál si mezi sebou vymění potřebná data. Karta se zejména dozví informace o samotné platbě (zejména částka a měna) a informace o terminálu (např. stát, v němž se nachází). Terminál se naopak dozví informace o kartě (zejména identifikátor karty a její certifikáty).
2. Autentizace karty. Na základě údajů poskytnutých kartou provede terminál její autentizaci. V současné době přichází v úvahu pouze autentizace typu DDA (viz dále).
3. Autentizace držitele karty. V praxi se nejčastěji lze setkat s variantou bez autentizace držitele a varianta autentizace pinem. U poslední zmíněné varianty existují dvě podvarianty. Buď pin zkontroluje mikropočítač karty, nebo klientova banka BK.
4. Analýza rizik. Na základě vyměněných dat (např. výše částky a stát, v němž se terminál nachází) provedou terminál i karta analýzu rizik. Podle závěrů svých analýz se dohodnou na jedné ze tří možností. Těmito možnostmi jsou zrušení transakce, autonomní transakce a transakce s povolením banky klienta.
5. Platba. Podle výše sjednané možnosti dojde buď ke zrušení transakce (tj. nedojde k platbě), nebo terminál od karty obdrží platební závazek *PZ*, na jehož základě si pak může po bance klienta nárokovat zaplacení sjednané částky. Tento platební závazek karta vygeneruje buď autonomně, nebo na základě povolení klientovy banky.

Popsaný princip protokolu si nyní budeme ilustrovat na dvou obvyklých scénářích. Prvním scénářem je platba malou částkou. V takovémto případě se nevyžaduje autentizace držitele karty a ani se nevyžaduje povolení klientovy banky. V případě platby větší částkou se už autentizace klienta provádí a vyžaduje se i povolení banky.

Scénář platby malou částkou ilustruje obr. 4.10.



Obrázek 4.10: Protokol pro platbu malou částkou

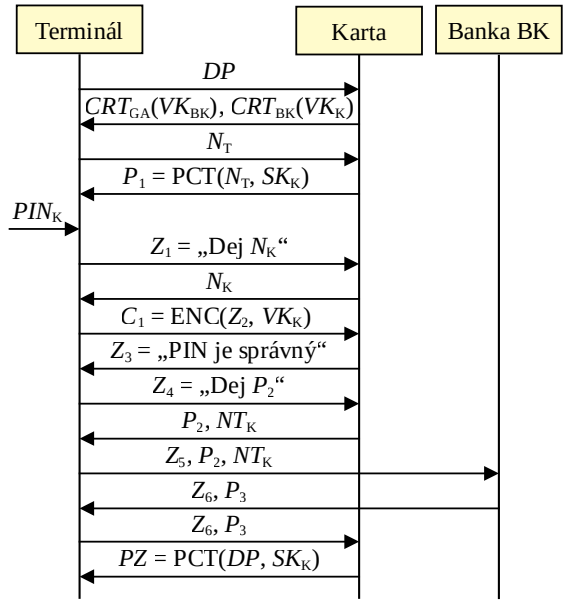
V prvním kroku protokolu zašle terminál T kartě K data o platbě *DP*. V nich je zejména uvedena částka k platbě a účet prodejce. Následně karta zašle terminálu certifikát $CRT_{GA}(VK_{BK})$ veřejného klíče banky spolu s certifikátem $CRT_{BK}(VK_K)$ veřejného klíče karty. Terminál si nejprve z kořenového certifikátu $CRT_{GA}(VK_{GA})$, který má uložen ve své paměti, zkontroluje autentičnost veřejného klíče VK_{GA} globální autority a jeho prostřednictvím pak ověří z certifikátu $CRT_{GA}(VK_{BK})$ identitu klientovy banky BK a autentičnost jejího veřejného klíče VK_{BK}. V této souvislosti připomínáme, že certifikát $CRT_{GA}(VK_{BK})$ je prakticky tvrzení, že „BK má VK_{BK}“, které je podepsáno globální autoritou. Terminál nakonec pomocí autentizovaného klíče VK_{BK} zjistí z certifikátu $CRT_{BK}(VK_K)$ identifikátor karty ID_K a ověří i autentičnost veřejného klíče VK_K karty.

Terminál nyní kartě zašle náhodné číslo N_T. Ta jej podepíše svým soukromým klíčem a vytvořený podpis $P_1 = \mathrm{PCT}(N_T, SK_K)$ zašle zpět terminálu. Ten jej pomocí veřejného klíče karty VK_K ověří. Pokud autentizační indikátor $W = \mathrm{VER}(N_T, P_1, VK_K) = 1$, tak je podpis platný a terminál má nyní potvrzeno, že protistranou je skutečně karta, která byla vydána bankou BK a má identifikátor ID_K. Popsaná autentizace karty se označuje zkratkou DDA („Dynamic Data Authentication“).

Terminál analýzou dat z dosavadního běhu protokolu došel k názoru, že autentizace držitele není nutná a odesláním zprávy Z_1 = „Žádost o *PZ*“ kartě nabídne přeskočit fázi autentizace držitele a přejít hned k platbě, a to autonomním způsobem. Pokud s tím karta souhlasí, tak terminálu zašle platební závazek *PZ*, což je prakticky její podpis dat o platbě. Platí tedy, že $PZ = \mathrm{PCT}(DP, SK_K)$. Terminál pomocí veřejného klíče karty přijatý podpis ověří a ukončí protokol. Prodejci pak na svém displeji indikuje, že platba proběhla v pořádku.

Terminál za určité období zasílá své bance pro každou jím uskutečněnou transakci trojici (ID_K, DP, PZ). Banka prodejce podle ID_K identifikuje klientovu banku a danou trojici jí přepošle. Banka klienta pak nejprve podle ID_K zjistí ze své databáze veřejný klíč VK_K karty a ověří, zda W = VER(DP, PZ, VK_K) = 1. Pokud tomu tak je, tak se držitel karty k uvedené platbě zavázal. Banka BK z dat DP o platbě zjistí požadovanou částku a číslo účtu prodejce. Tuto částku pak odebere z účtu držitele karty a převede ji na účet prodejce. Nevýhodou popsaného scénáře je, že kartou může platit útočník, který ji klientovi předtím ukradl (nevyžaduje se autentizace držitele karty) a že klient nemusí mít na svém účtu dost peněz k zaplacení poskytnutého zboží či služby (nevyžaduje se povolení banky).

Scénář platby velkou částkou ilustruje obr. 4.11. Zahájení protokolu a autentizace karty jsou stejné jako v předchozím případě. V prvním kroku protokolu zašle terminál T kartě K data o platbě DP. Následně karta zašle terminálu certifikát $CRT_{GA}(VK_{BK})$ veřejného klíče banky spolu s certifikátem $CRT_{BK}(VK_K)$ veřejného klíče karty. Terminál si pomocí veřejného klíče VK_{GA} globální autority ověří z certifikátu $CRT_{GA}(VK_{BK})$ identitu klientovy banky BK a autentičnost jejího veřejného klíče VK_{BK}. Nakonec pak z certifikátu $CRT_{BK}(VK_K)$ zjistí identifikátor karty ID_K a ověří i autentičnost jejího veřejného klíče VK_K. Terminál nyní kartě zašle náhodné číslo N_T. Ta jej podepíše svým soukromým klíčem a vytvořený podpis P_1 = PCT(N_T, SK_K) zašle zpět terminálu. Ten jej pomocí veřejného klíče karty VK_K ověří. Pokud je podpis platný, tak má terminál nyní potvrzeno, že protistranou je karta, která byla vydána bankou BK a má identifikátor ID_K.



Obrázek 4.11: Protokol pro platbu velkou částkou

Dejme tomu, že karta i terminál mají nastavena kritéria pro analýzu rizik tak, že se shodnou na nutnosti autentizovat klienta prostřednictvím karty a na nutnosti vyžádat si povolení banky. Nejprve proběhne autentizace klienta. Klient je na displeji terminálu vyzván k tomu, aby pomocí klávesnice terminálu zadal svůj pin PIN_K. Po zadání pinu terminál zašle kartě zprávu Z_1, což je žádost o zaslání náhodného čísla N_K. Karta požadované náhodné číslo vygeneruje a pošle je terminálu. Ten z něho a pinu vytvoří zprávu $Z_2 = (PIN_K \parallel N_K)$, kterou pak pomocí veřejného klíče karty zašifruje do podoby kryptogramu $C_1 = \mathrm{ENC}(Z_2, VK_K)$. Tento kryptogram odešle kartě. Zahrnutím náhodného čísla N_K do procesu autentizace se zajistí, že kryptogram C_1 bude pokaždé jiný a případný útočník se tak nebude moci vůči kartě autentizovat kryptogramem, který před krádeží karty zachytil při některé předchozí úspěšné autentizaci klienta. Karta následně kryptogram C_1 svým soukromým klíčem dešifruje, čímž získá $\mathrm{DEC}(C_1, SK_K) = (PIN_K \parallel N_K)$. Zkontroluje správnost náhodného čísla N_K a hodnotu pinu porovná s hodnotou uloženou ve své paměti. V případě shody odešle terminálu zprávu Z_3 = „*PIN je správný*“. Tím skončila autentizace klienta a pokračuje se získáním povolení od klientovy banky.

Terminál i karta si z vyměněných údajů sestrojí zprávu $Z_5 = (ID_K \parallel DP)$, která je určena pro klientovu banku. Tato zpráva obsahuje základní údaje o transakci, jimiž jsou identifikátor karty ID_K a data DP o platbě. Terminál následně zašle kartě zprávu Z_4, což je žádost o zaslání pečeti pro zprávu Z_5. Karta pomocí relačního klíče RK_K vypočítá pro tuto zprávu pečeť $P_2 = \mathrm{PCT}(Z_5, RK_K)$. Připomínáme, že relační klíč karta odvodí z hlavního klíče HK_K karty a počtu NT_K doposud uskutečněných transakcí podle vztahu $RK_K = \mathrm{ODF}_2(HK_K, NT_K)$, přičemž ODF_2 je stanovená odvozovací funkce. Karta zašle vypočítanou pečeť P_2 spolu s číslem NT_K terminálu. Ten trojici (Z_5, P_2, NT_K) zašle své bance, která ji pak prostřednictvím platební sítě přepošle bance klienta.

Banka klienta ze zprávy $Z_5 = (ID_K \parallel DP)$ zjistí základní data o transakci. Pomocí identifikátoru ID_K a svého hlavního klíče HK_{BK} nejprve sestrojí hlavní klíč karty $HK_K = \mathrm{ODF}_1(HK_{BK}, ID_K)$ a poté pomocí doručeného počtu transakcí NT_K zjistí i relační klíč karty $RK_K = \mathrm{ODF}_2(HK_K, NT_K)$. Prostřednictvím relačního klíče nyní může ověřit správnost pečeti. Pokud $W = \mathrm{VER}(Z_5, P_2, RK_K) = 1$, tak autorem zprávy $Z_5 = (ID_K, DP)$ je karta s identifikátorem ID_K. Pokud je požadovaná platba možná, tak banka BK na účtu klienta s kartou ID_K rezervuje požadovanou částku. Následně banka vytvoří dvojici (Z_6, P_3), kde Z_6 je zpráva pro terminál a kartu, že transakce je schválena, resp. zamítnuta a pečeť $P_3 = \mathrm{PCT}(P_2 \parallel Z_6, RK_K)$ zaručuje autentičnost zprávy Z_6 a přes pečeť P_2 i její vazbu na zprávu Z_5.

Dvojici (Z_6, P_3) banka BK zašle přes platební síť a banku prodejce k terminálu. Terminál se ze zprávy Z_6 dozví rozhodnutí banky BK ohledně schválení, resp. zamítnutí probíhající transakce a dvojici (Z_6, P_3) předá kartě. Ta ověří správnost pečeti P_3 kontrolou, zda $\mathrm{VER}(P_2 \parallel Z_6, P_3, RK_K) = 1$. Pokud tomu tak je a zároveň banka transakci povolila, tak karta odešle terminálu platební závazek $PZ = \mathrm{PCT}(DP, SK_K)$. Terminál pomocí veřejného klíče karty VK_K přijatý podpis ověří a protokol ukončí. Prodejci pak na svém displeji indikuje, že transakce proběhla v pořádku. Samotný převod peněz z účtu klienta na účet prodejce probíhá stejně jako v předešlém scénáři.

Literatura

Literatura

[1] Burda K.: Úvod do kryptografie. Brno, CERM 2005.

[2] Burda K.: Diffie-Hellman Protocol as a Symmetric Cryptosystem. International Journal of Computer Science and Network Security, roč. 2018, č. 7, s. 33-36.

[3] Ramsdell B., Turner S.: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751, Internet Engineering Task Force, Fremont 2010.

[4] Kaufman C. aj.: Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, Internet Engineering Task Force, Fremont 2014.

[5] Arends R. aj.: DNS Security Introduction and Requirements. RFC 4033, Internet Engineering Task Force, Fremont 2005.

[6] Rescorla E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, Internet Engineering Task Force, Fremont 2018.

[7] Ylonen T., Lonvick C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251, Internet Engineering Task Force, Fremont 2006.

[8] Ylonen T., Lonvick C.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, Internet Engineering Task Force, Fremont 2006.

[9] Ylonen T., Lonvick C.: The Secure Shell (SSH) Authentication Protocol. RFC 4252, Internet Engineering Task Force[24], Fremont 2006.

[10] Ylonen T., Lonvick C.: The Secure Shell (SSH) Connection Protocol. RFC 4254, Internet Engineering Task Force, Fremont 2006.

[11] Kent S., Seo K.: Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, Fremont 2005.

[12] Kent S.: IP Authentication Header. RFC 4302, Internet Engineering Task Force, Fremont 2005.

[13] Kent S.: IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, Fremont 2005.

[14] Burda K.: Anonymizační síť Tor. Elektrorevue, roč. 2015, č. 4, s. 124-135.

[15] Dingledine R., Mathewson N.: Tor Protocol Specification. The Tor Project. Cambridge 2008.

[16] -: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2016. IEEE Computer Society. N.York 2016.

[17] Harkins D., Kumari W: Opportunistic Wireless Encryption. RFC 8110, Internet Engineering Task Force, Fremont 2017.

[18] -: Media Access Control (MAC) Security. IEEE Std 802.1AE-2018. IEEE Computer Society. N.York 2018.

[19] -: Port-Based Network Access Control. IEEE Std 802.1X TM -2010. IEEE Computer Society. N.York 2010.

[20] Franks J. aj.: HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, Internet Engineering Task Force, Fremont 1999.

[21] Aboba B. aj.: Extensible Authentication Protocol (EAP). RFC 3748, Internet Engineering Task Force, Fremont 2004.

[22] -: IEEE Standard for Local and metropolitan area networks. Port-Based Network Access Control. IEEE 802.1X-2010, IEEE, N. York 2010.

[23] Neuman C. aj.: The Kerberos Network Authentication Service (V5). RFC 4120, Internet Engineering Task Force, Fremont 2005.

[24] Sakimura N. aj.: OpenID Connect Core 1.0. OpenID Foundation, San Ramon 2014.

[25] Hardt D.: The OAuth 2.0 Authorization Framework. RFC 6749, Internet Engineering Task Force, Fremont 2012.

[26] Rigney C. aj.: Remote Authentication Dial-In User Service (RADIUS). RFC 2865, Internet Engineering Task Force, Fremont 2000.

[27] Burda K.: Základy elektronických zabezpečovacích systémů. CERM, Brno 2017.

[28] -: EMV. 3-D Secure. Protocol and Core Functions Specification. EMVCo, Mountain View 2018.

[29] Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System. 2009. Dostupné na: https://bitcoin.org/bitcoin.pdf

[30] -: EMV. Integrated Circuit Card. Specifications for Payment Systems. Book 2. Security and Key Management. Version 4.3. EMVCo, Mountain View 2011.

KRYPTOGRAFIE OKOLO NÁS

Karel Burda

Vydavatel:
CZ.NIC, z. s. p. o.
Milešovská 5, 130 00 Praha 3
Edice CZ.NIC
www.nic.cz

1. vydání, Praha 2019
Kniha vyšla jako 24. publikace v Edici CZ.NIC.
Tisk: H.R.G. spol. s r.o., Svitavská 1203, 570 01 Litomyšl
Sazba: Tomáš Brejcha

ISBN 978-80-88168-49-2 (tištěná verze)
ISBN 978-80-88168-50-8 (ve formátu EPUB)
ISBN 978-80-88168-51-5 (ve formátu MOBI)
ISBN 978-80-88168-52-2 (ve formátu PDF)

Edice CZ.NIC

O knize Kniha, kterou držíte v rukou, popisuje využití kryptografie v běžném životě. Nejprve jsou v knize vysvětleny vlastnosti vybraných kryptografických funkcí, jako je šifrování, hešování a podobně. S pomocí těchto obecných funkcí jsou poté popsány přenosové protokoly PGP, TLS, WPA, IPsec, SSH, Tor, DNSsec, IKE a MACsec, autentizační protokoly DAA, EAP, Kerberos a OpenID Connect, autorizační protokol OAuth, přístupový protokol RADIUS a nakonec platební protokoly 3D Secure a Bitcoin. Navíc jsou v knize popsány i systémy elektronické kontroly vstupu, elektronické bankovnictví a protokoly platebních karet.

O autorovi Karel Burda vystudoval v roce 1981 Vysokou vojenskou technickou školu v L. Mikuláši. Po praxi u vojsk přednášel na své alma mater a později na Vojenské akademii v Brně. V současné době přednáší na Vysokém učení technickém v Brně. Specializuje se na kryptografii, komunikační bezpečnost a fyzickou bezpečnost. Je autorem knih Aplikovaná kryptografie, Úvod do kryptografie a Základy elektronických zabezpečovacích systémů.

O edici Edice CZ.NIC je jednou z osvětových aktivit správce české národní domény. Ediční program je zaměřen na vydávání odborných, ale i populárně naučných publikací spojených s Internetem a jeho technologiemi. Kromě tištěných verzí vychází v této edici současně i elektronická podoba knih. Ty je možné najít na stránkách knihy.nic.cz.

knihy.nic.cz

ISBN 978-80-88168-52-2

9 788088 168522