

Off-the-record protokol pro výměnu textových zpráv

Off-the-record Protocol for exchanging
text messages

Jiří Vávra



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jiří VÁVRA**
Osobní číslo: **A09663**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Off-the-record protokol pro výměnu textových zpráv**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Popište principy šifrování pomocí OTR.
3. Zjistěte současný stav užívání OTR.
4. Napište knihovnu, případně aplikaci OTR protokolu v C++ nebo JavaScriptu.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **VONDRUŠKA, Pavel. Kryptologie, šifrování a tajná písma. Albatros, 2006. ISBN 8000018888.**
2. **MOLDOVYAN Nick, MOLDOVYAN Alex. Innovative Cryptography, Second Edition. Cengage Charles River Media, 2007. ISBN 9781584504672.**
3. **DENIS, Tom St. Cryptography for Developers. Syngress Publishing. 2007. ISBN 9781597491044.**
4. **PRATA, Stephen, Mistrovství v C++, 3., aktualiz. vyd., Computer press, 2007, ISBN 9788025117491.**
5. **GOLDBERD, Ian et al. Off-the-Record Messaging [online]. [cit. 2012-06-02]. Dostupné z: <http://www.cypheerpunks.ca/otr/>**

Vedoucí bakalářské práce:

Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

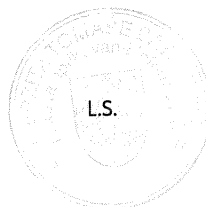
24. února 2012

Termín odevzdání bakalářské práce:

8. června 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

V této práci se budeme zabývat Off-the-record protokolem, neboli šifrováním komunikace způsobem, které nám jiné šifrovací prostředky nenabídnou – šifrovanou komunikaci s ověřenou druhou stranou, ale přesto s vlastností nedokazatelnosti vlastnictví zprávy.

Navíc ve vypracované aplikaci v jazyku JavaScript ukážeme základní principy Off-the-record protokolu v praxi – výměnu klíčů na základě Diffie-Hellman protokolu, podepisování zpráv pomocí RSA a princip perfect forward secrecy a repudiability.

Klíčová slova: Off-the-record, OTR, komunikace, šifrování

ABSTRACT

In this thesis we are going to describe Off-the-record protocol. To encrypt our communication in way, which other cryptographical application cannot offer – encrypted communication with authenticated partner. But still the ownership of any message cannot be proven.

Moreover in application written in Javascript will be demonstrated basic principles of Off-the-record protocol – key exchange using Diffie-Hellman protocol, signing messages using RSA a principles of perfect forward secrecy and repudiability.

Keywords: Off-the-record, OTR, messaging, cryptography

- Do you have an idea for your story yet?
- No, I'm waiting for inspiration. You can't just turn on creativity like a faucet. You have to be in the right mood.
- What mood is that?
- Last-minute panic.

Calvin and Hobbes

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

- o že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....

Podpis diplomanta

Obsah

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 Historie Šifrování	13
1.1 Steganografie	13
1.1.1 FYZICKÁ STEGANOGRAFIE	13
1.1.2 DIGITÁLNÍ STEGANOGRAFIE	14
1.2 Kryptografie	14
1.2.1 KLASICKÁ KRYPTOGRAFIE	15
1.2.2 MODERNÍ KRYPTOGRAFIE	15
Data encryption Standard (DES)	16
RSA	16
Pretty Good Privacy (PGP)	16
Kvantová kryptografie	17
2 Prostředky a principy týkající se OTR protokolu	18
2.1 Diffie-Hellman	18
2.2 Message Authentication Code (MAC)	19
2.3 Advanced Encryption Standard (AES)	19
2.4 Secure Hash Algorithm (SHA)	20
2.5 Socialist Millionaires' protokol	20
2.6 Man-in-the-middle útok	21
3 Popis OTR	22
3.1 OTR v.1	22
3.1.1 ŠIFROVÁNÍ	22
3.1.2 ZAPOMÍNÁNÍ KLÍČŮ	23
3.1.3 AUTENTIZACE	23
3.1.4 ODHALENÍ MAC KLÍČŮ	24
3.1.5 ÚTOK NA OTR v.1	24

3.2	OTR v.2	25
3.2.1	MODIFIKOVANÝ SOCIALIST MILLIONAIRES' PROTOKOL.....	25
3.2.2	ŠIFROVÁNÍ.....	25
4	Aktuální použití OTR	28
4.1	IM komunikátory	28
4.2	Knihovny	28
4.3	OTR přes SMS.....	29
II	PRAKTICKÁ ČÁST	30
5	Použitá řešení třetích stran	33
5.1	BigInt	33
5.2	CryptoJS	33
6	Aplikace v JavaScript	34
6.1	DH autentizace účastníků a výměna sdíleného tajemství.....	34
6.2	Messenger	37
	ZÁVĚR	41
	CONCLUSIONS	42
	SEZNAM POUŽITÉ LITERATURY	42
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	46
	SEZNAM OBRÁZKŮ	47
	SEZNAM PŘÍLOH	48

ÚVOD

Potřeba komunikaci zamaskovat či znemožnit její přečtení se datuje již od dávných dob. Stručná historie vývoje je k nalezení v sekci 1 této práce. S rozvojem technologií se rozšiřují nejen možnosti, jak spolu komunikovat, ale také jak tuto komunikaci odposlouchávat a následně ji zneužít. Většina technologií zabývajících se použitím kryptografie se specializuje na různé aspekty komunikace: zamezení přečtení obsahu třetí straně či ověřování totožností komunikujících stran. To je samozřejmě důležité při obchodním styku. Ale i v soukromé komunikaci stále stoupá na významu potřeba chránit své soukromí a dbát na „dobré jméno.“

Běžný člověk tráví spoustu času na Internetu a pochopitelně skrz něj i komunikuje. Stále ještě však není pravidlem, aby zprávy byly vyměňovány po zabezpečených kanálech. I přesto na nás nečíhá nebezpečí pouze ve virtuálním světě Internetu, ale i u nás doma, v baru či kavárně. Zlodějem, který nám ukradne počítač, virem, trojanem, hackerem zneužívajícím ještě nezaplátovanou díru v systému, ztrácíme kontrolu nad naším majetkem, daty. Útočník tedy dostává do svých rukou data, která jsme vytvořili a slova, která jsme kdy v rozhovorech vyřkli. Ale může se to obrátit proti nám nejen v soukromých záležitostech, ale i tváří v tvář státnímu aparátu. Spousta moderních kryptografických prostředků jako PGP, či S/MIME nám sice zajistí neodposlouchávatelnou komunikaci a jistotu o identitě našeho protějšku, ale v případě odcizení dat dostává útočník pádné důkazy. Tyto požadavky motivují k hledání následujících vlastností:

- Šifrování komunikace: Nikdo třetí nebude schopen přečíst konverzaci.
- Perfect forward secrecy: Prolomením šifrovacího klíče třetí stranou není zkompromitována naše předchozí komunikace.
- Autentizace účastníků: Účastníci komunikace jsou si jisti, že druhá strana je opravdu ta, za kterou se vydává.
- Deniable authentication: Při úniku naší komunikace (získáním lokálních dat) je neprůkazné vlastnictví výroku.

Splnění těchto požadavků nám zajistí Off-the-record (OTR) protokol. Principiálně se to dá představit, že komunikujeme v uzavřené odhlučněné místnosti – nikdo vás neslyší a máte jistotu, že mluvíte s kým chcete. Ale po ukončení komunikace a opuštění místnosti můžete všem říkat, o čem jste si povídali, ale nemůžete to dokázat.

OTR protokol používá již stávajících šifrovacích prostředků a vhodně je zkombinuje, aby dosáhl svého cíle. Tyto prostředky jsou popsány v kapitole 2. Vlastní princip OTR protokolu je popsán v kapitole 3. Nasazení OTR v praxi je zkoumáno v kapitole 4. V druhé části této práce je představena jednoduchá aplikace OTR v JavaScriptu.

I. TEORETICKÁ ČÁST

Kapitola 1

Historie Šifrování

V této kapitole bude vysvětlená stručná historie šifrování. To jak z pohledu ukrývání informace – steganografie [16], tak z pohledu změny zprávy do, na první pohled, nečitelné podoby – kryptografie [4] [23].

1.1 Steganografie

Slovo steganografie pochází ze spojení řeckých slov steganos (schovaný) a graphein (psát). Jedná se o ukrytí zprávy před zraky ostatních lidí. Během staletí se tato metoda rozvinula ve všech částech světa nezávisle na sobě.

1.1.1 Fyzická steganografie

Na to, jak ukrýt zprávu, je spousta nápadů. Například v Číně napsali zprávu na hedvábný papír, zmačkali do kuličky, zalili voskem a posel pak tuto kuličku spolkl. Jinou možností bylo vytetovat zprávu na vyholenou hlavu otroka a pak ji nechat zase zarůst. Až posel došel na místo určení, oholili mu hlavu a mohli si přečíst zprávu. V překažení útoku na Řecko Perskou říší zase pomohl text vyrytý na dřevěnou desku, která byla poté překryta vrstvou vosku. Tak se deska jevila jako prázdná, nepoužitá a posel tak mohl projít nepovšimnut skrz hlídky.

Velmi oblíbené byly různé neviditelné inkousty. Postupně se od jednoduchých inkoustů, které byly viditelné po zahřátí, vyvinuly metody sofistikovanějších sloučenin, které potřebovaly specifické chemikálie, aby se ukázaly. K dokonalosti přivedl neviditelný inkoust pravděpodobně italský vědec G. Porta, který v 16. století popsal způsob, jak pomocí roztoku ledku a octa napsat zprávu na bílek skrz skořápku uvařeného vejce. Zpráva tak vyšla najevo po oloupaní vejce. Neviditelné inkousty však ztratily na významu, když se podařilo vyvinout univerzální vývojku, která dokázala ukázat neviditelný text na základě změn na vláknech papíru, které vznikly při kontaktu s neviditelným inkoustem.

Další možností bylo skrytí tajné zprávy do jiného textu. Existuje spousta možností, např. prosté čtení každého druhého písmene každého slova v textu (bylo použito za

druhé světové války). Za druhé světové války Němci také vyvinuli technologii mikroteček. Ačkoliv měly velikost běžného interpunkčního znaménka, dokázaly nést informaci, která se vejde na běžnou stránku, včetně nákresů a fotografií. První mikrotečka byla objevena u německého agenta na nadepsané obálce v roce 1914.

Při použití tiskáren může být významné slovo označeno třeba použitím delšího odsazení od předešlého slova. Používá se také označení dokumentu vytištěním špatně čitelných žlutých teček, které obsahují např. informaci o tiskárně, která dokument vytiskla.

1.1.2 Digitální steganografie

Ačkoliv se po vynalezení moderních šifer a rozšíření Internetu začalo používat více šifrování, než steganografie, v poslední době steganografie zase začíná nabývat na významu. Částečně kvůli tomu, že spousta států (a mezinárodních úmluv) chce nebo už dokonce praktikuje legálními prostředky omezení šifrování. Jedná se například o omezení maximální síly šifry, povinnost rozšifrovat svůj disk na hranicích nebo třeba zavedení presumpce viny při odmítnutí poskytnutí přístupu ke svým datům. Navíc se steganografie uplatňuje při ochraně autorských práv, kdy se do digitálního souboru začlení informace o autorství (watermark).

Oblíbená aplikace je vložení zprávy nebo obrázku do nosného obrázku. Jako příklad poslouží nosný obrázek o rozlišení 1024x768 pixelů o barevné hloubce 24 bitů. Jeden pixel je tudíž tvořen třemi byty (8 bitů na každý barevný kanál RGB). Při použití nejméně významného bitu (least significant bit) v každé barvě můžeme do každého pixelu obrázku skrýt tři bity. To v tomto případě v cca 2,4 MB skryje 300 KB tajných dat.

Za způsob steganografie by se dal považovat i skrytý oddíl při šifrování disku pomocí programu TrueCrypt. Při nutnosti někomu ukázat obsah zašifrovaného disku se zadá heslo, při kterém se odemkne část disku, kde mohou být uloženy nějaké soubory fungující jako kamufláž. Citlivá část má však vlastní heslo a tak zůstane skryta. Navíc nelze ani jednoduše poznat, že zveřejněný oddíl obsahuje ještě nějakou jinou skrytou část s daty.

1.2 Kryptografie

Slovo kryptografie pochází z řeckých slov krypto (tajný) a graphein (psát). Samotné šifrování je staré jako písmo samo, v podstatě i písmo je šifrování. Zvláště mayská kultura dokázala svým písemným systémem dlouho odolávat snahám o rozluštění. Například text neběžel pouze jedním směrem (např. zleva doprava), pro vyjádření jedné slabiky nepoužívali jen jeden obrázek a dokonce mohli na jednu pozici "napsat" slabik víc.

1.2.1 Klasická kryptografie

Mezi nejstarší známé šifry patří hebrejský atbash – jednoduchá substituční šifra pomocí otočené abecedy, v Řecku se používala substituční šifra a podle Caesara se dokonce pojmenovala jednoduchá substituční šifra. V kámásútře je zmínka o šifrování mezi uměními.

Ve středověku měla náskok arabská kultura před evropskou. Jejich největším průlomem byl vynález kryptoanalýzy na základě frekvenční analýzy. Ve středověku se také vyvinula metoda polyalfabetické substituce – šifrování podle hesla. Její vrchol se označuje Vigenèrova šifra pojmenovaná podle francouzského diplomata B. de Vigenère, který ji dotáhl do konečné podoby.

Opravdu významnou roli začala mít kryptografie za válek vedených ve 20. století. Díky vynálezu rádia mohly být informace předávány na velkou vzdálenost v reálném čase. Ale tento způsob je velice náchylný na odposlouchávání. Ze začátku se používaly hlavně šifry založené na složitějších substitucích nebo za použití tzv. kódových knih. Za druhé světové války se začaly používat stroje. Asi nejznámějším mechanickým šifrovacím přístrojem byla německá Enigma, pracující na principu propojených rotujících disků. Tím vytvářela velice složitou substituční šifru. Američané používali kód Navajo – je to pseudošifra, kdy příslušníci indiánského kmene Navajo používali svůj jazyk, který je velice odlišný od jakéhokoliv jiného.

Jedinou opravdu bezpečnou šifrou je Vernamova šifra (označována také jako one-time pad), která byla patentována G. Vernamem v roce 1917. Je to substituční šifra, kdy je použit stejně dlouhý klíč, jako je délka zprávy. Klíč musí být náhodný a nesmí se nikdy opakovat. Potom, i kdyby útočník měl k dispozici neomezené výpočetní prostředky na luštění, nikdy se nemůže z principu dobrat původního textu.

1.2.2 Moderní kryptografie

S rozvojem počítačů se mohlo začít používat výpočetně náročné šifrování. V posledních několika desetiletí vzniklo velké množství norem a algoritmů, zde si představíme to nejvýznamnější, co vzniklo.

Symetrické šifry používají klíč, který sdílí obě strany – používá se jak při šifrování tak dešifrování. Nemohou se tedy setkat na nezabezpečeném kanálu dvě strany, které by mohli začít šifrovaně komunikovat. Jejich výhodou je však mnohem větší rychlost. Asymetrické naopak pracují s veřejnými klíči, které mohou být veřejně distribuovány. I když neprovádí tolik matematických úkonů jako symetrické šifry, tyto úkony jsou natolik náročně, že jejich rychlost je zlomková.

Advanced Encryption Standard (AES) je použit v OTR, proto se dá nalézt v sekci 2.3

Data encryption Standard (DES)

Tento standard blokové symetrické šifry vyvinula firma IBM a v roce 1976 se přijal za standard. Původní účel byl pro transfer bankovních dat a tento algoritmus měl být implementován hardwarově na malém čipu. K šifrování se používá 64 bitový klíč (z kterých pouze 56 je efektivních, zbytek je kontrolních). Samotný algoritmus používá substitute a permutace (tzn. nahrazení bitové hodnoty jinou a záměna pořadí bitů v bloku). Jeho velkou slabinou je hlavně „délka“ jeho klíče, takže v zájmu zachování bezpečnosti byl tento standard rozšířen o další varianty, které bezpečnost zvyšují. Nejpoužívanější je Triple DES.

Triple DES aplikuje na zprávu třikrát DES hned po sobě a používá klíč o délce 168 bitů (3·56). Variantou je třeba EDE, při které se zpráva prvně zašifruje, pak dešifruje a znova zašifruje. Každý z těchto kroků je samozřejmě proveden s jiným (56 bitovým) klíčem.

Asymetrické šifrování je založeno na jednosměrné funkci. To znamená, že výpočet jedním směrem je jednoduchý, ale druhým takřka nemožný. Jako příklad z běžného života se udává třeba míchání barev, kdy namíchat barvu je velice jednoduché, ale zjistit z kterých dvou složek jsme tu barvu namíchali je prakticky nezjistitelné. V počítačovém světě se za jednosměrné funkce považují např. umocňování dvou velkých čísel. První úspěšná implementace byla Diffie-Hellman protokol, viz 2.1. Mezi asymetrické metody patří i hashovací funkce, jako např. SHA, viz 2.4.

RSA

Šifra RSA má název podle iniciálů autorů – Rivest, Shamir, Adleman. Vznikla v roce 1977 a jde o první algoritmus, který se hodí jak na šifrování tak na digitální podepisování. Její princip je, že faktorizace (rozdělení na prvočísla) čísla $n = p \cdot q$ je výpočetně velice náročná úloha. Alice si vytvoří pár soukromého a veřejného klíče. Veřejný klíč zveřejní a pomocí něj jí může kdokoliv poslat šifrovanou zprávu, kterou Alice rozluští pomocí svého soukromého klíče. U podpisu se používá princip opačný princip. Hash zprávy Alice „dešifruje“ pomocí svého soukromého klíče. Příjemce potom ověří, že digitální podpis odpovídá veřejnému klíči Alice tak, že „zašifruje“ tento podpis pomocí veřejného klíče. Pak vypočítá hash zprávy a pokud hodnota odpovídá tomu, co dostal od Alice, přišla zpráva nepozměněna.

Pretty Good Privacy (PGP)

Spojením výhod symetrických a asymetrických šifer vzniklo PGP. Je to vynález P. Zimmermanna a první verze byla uvolněna v roce 1991. Později bylo PGP standardizováno pro internetové použití a uveřejněno pod názvem OpenPGP. Používá se i na digitální podepisování. Princip posílání zpráv pomocí PGP je následující:

Alice:

- Zpráva se zkomprimuje a zašifruje pomocí symetrického klíče.
- Symetrický klíč se zašifruje pomocí asymetrického Bobova veřejného klíče.
- Pošle se zpráva obsahující zašifrovanou zprávu (symetricky) a zašifrovaný klíč (asymetricky).

Bob:

- Přijme zprávu a rozdělí ji na šifrovanou zprávu a šifrovaný klíč.
- Rozšifruje symetrický klíč svým soukromým asymetrickým klíčem.
- Rozšifruje zprávu symetrickým klíčem.

Kvantová kryptografie

Při vzniku kvantových počítačů by mohla padnout bezpečnost všech dosavadních šifer (samozřejmě až na Vernamovu). Zatímco výroba použitelného kvantového počítače je někde v daleké budoucnosti, již nyní můžeme použít kvantovou kryptografii. Tento druh kryptografie nespolehá na matematickou náročnost, ale na fyzikální zákony.

Myšlenku poprvé rozvinul v 60. letech 20. století S. Wiesner, který navrhl kvantové peníze. Bohužel byl na tu dobu příliš pokrokový a tak první kvantový kryptografický protokol byl navrhnut v roce 1984. Principem je, že Alice vytváří klíč pomocí polarizovaných fotonů náhodně ve směrech 0° , 45° , 90° , nebo 135° . Bob náhodně nastaví měření kolmé nebo diagonální báze a registruje výsledky měření. Po skončení vysílání Bob řekne Alici, jaký typ pro daný foton použil a Alice mu řekne, jaké byly správně. Ty se poté převedou na bity a vytvoří tak klíč. Ze své podstaty tento systém nemůže být odposloucháván, protože útočník nemá šanci zjistit, v jaké přesné polarizaci byly fotony vysílány a tak je nemůže přeposlat Bobovi aniž by Bob s Alicí detekovali útočníka.

Kapitola 2

Prostředky a principy týkající se OTR protokolu

V této sekci si představíme prostředky použité v OTR protokolu v rozsahu, který je potřeba pro pochopení mechanismu v OTR použitém. Stejně tak je zde uveden i systém tzv. Man-In-The-Middle útoku. **Upozornění: všude, kde se bude vyskytovat g^x , $(g^x)^y$ a pod. je myšleno $g^x \bmod p$ atd.,** kde p je velké prvočíslo. Činím tak z důvodu lepší přehlednosti textu.

2.1 Diffie-Hellman

Tento protokol byl vynalezen v roce 1976 Whitfieldem Diffiem a Martinem Hellmanem [10]. Jedná se o první implementaci šifrování za pomoci dvou klíčů.

Podle nich každý účastník zveřejní svou „adresu“ a šifrovací klíč. Algoritmus šifrování je veřejný, takže pokud tedy chce někdo poslat šifrovanou zprávu, vezme ji a zašifruje pomocí jednocestné šifrovací funkce pomocí veřejně dostupného klíče. Vzhledem k tomu, že pouze adresát má k dispozici potřebnou informaci na rozšifrování (tzn. svůj soukromý dešifrovací klíč), vyplývá, že:

- Není potřeba žádného předchozího kontaktu mezi odesílatelem a adresátem. Lze jednoduše přijímat nové účastníky.
- Není potřeba mnoho klíčů. U asymetrického šifrování roste počet klíčů s počtem účastníků lineárně, u symetrického šifrování kvadraticky.
- Veškeré informace potřebné k přenosu zpráv mohou být vystaveny veřejně.

Pro potřeby OTR protokolu se spokojíme se základní implementací DH algoritmu: Potřebujeme prvočíslo p a generátor g grupy \mathbb{Z}_p^* . Alice a Bob si vyberou nějaká čísla x_A , resp. x_B (soukromé klíče). Vypočítají g^{x_A} , resp. g^{x_B} (veřejné klíče) a vymění si je přes nezabezpečený kanál. Alice a Bob si teď mohou spočítat sdílené tajemství $(g^{x_B})^{x_A} = g^{x_A x_B}$, resp. $(g^{x_A})^{x_B} = g^{x_A x_B}$. Toto sdílené tajemství se používá ke generování krátkodobých šifrovacích klíčů.

2.2 Message Authentication Code (MAC)

Při přijímání zprávy chceme vědět, zda zpráva přišla neporušená a autentická. Cílem tedy není zprávu zašifrovat, ale zabezpečit. Oblíbená MAC konstrukce je HMAC [3] založená na jednosměrné hashovací funkci.

Alice použije svou kopii MAC klíče k na spočítání hodnoty MAC své zprávy M algoritmem A a přiloží MAC ke své zprávě (pro OTR potřeby skrz zabezpečený kanál). Bob spočítá MAC přijaté zprávy za použití (sdíleného) klíče a ověří integritu a autenticitu zprávy. Čili MAC se spočítá $MAC = A_k(M)$.

Jak je vidět, stejný MAC dokáže vytvořit kdokoli, kdo zná klíč k , takže když Bob obdrží zprávu s validním MAC a uvědomí si, že ji neposílal on a pouze Alice zná tento sdílený klíč, tak tu zprávu poslala Alice. Pro potřeby OTR se navíc hodí, že MAC dokáže vytvořit kdokoli se znalostí klíče k , takže Bob nemůže dokázat, že tu zprávu poslala Alice – mohl ji vytvořit sám.

2.3 Advanced Encryption Standard (AES)

V roce 1997 Americký úřad pro standardizaci (NIST) začal hledat nástupce za DES. A ten je nazván AES. Bylo předloženo několik návrhů, z nichž byl vybrán algoritmus Rijndael [7] [8], pojmenovaný po autorech: J. Daemen a V. Rijmen. Tak v roce 2001 NIST vydal vyhlášení o novém standardu [21].

AES je symetrická bloková šifra s bloky o velikosti 128 bitů. Délka klíče může být dle potřeby 128, 192 nebo 256 bitů. Na délce klíče závisí počet provedených rund (10, 12, nebo 14). V implementaci OTR je AES v counter módu (AES-CTR) [14].

NIST určil celkově pět různých módů pro AES a další FIPS schválené šifry [13]. AES-CTR mění blokovou šifru AES ve streamovou. Encrypter použije inicializační vektor (IV) a counter (ten může být opravdu zastoupen posloupností, kde pro každý další blok přičte +1). Counter musí mít hlavně tu vlastnost, že bude unikátní a nebude použit vícekrát. Výsledek se zašifruje pomocí klíče a vznikne tak blok (O) zašifrovaných pseudonáhodných dat. Ten se pak pomocí XOR spojí s naší zprávou a vznikne nám zašifrovaná zpráva, kterou můžeme poslat. Můžeme říci, že výsledný blok O vznikne zašifrováním klíčem k nonce N (který je stabilní po celou dobu aktuální komunikace) a counteru C (ten se pro každý blok mění), tzn. $O = CRYPT_k(N, C)$. Výsledná šifrovaná zpráva M' vznikne ze zprávy M jako $M' = M \oplus O$. Výhodou je, že se dají bloky O předpočítat a následně aplikovat paralelně .

2.4 Secure Hash Algorithm (SHA)

NIST zveřejnil roku 1993 SHA standard ¹⁾. Je to jednosměrná funkce, která vytvoří ze zadaných dat výstup standardní bitové délky. Aktuální verze [22] vydána v březnu tohoto roku specifikuje výsledný hash o délce 160, 224, 256, 384, nebo 512 bitů.

Hashovací funkce h má dvě potřebné vlastnosti:

- Bezkoliznost:
 - Slabá bezkoliznost: pro zprávu M_1 je výpočetně nemožné najít takovou zprávu M_2 tak, že platí $h(M_1) = h(M_2)$.
 - Silná bezkoliznost: je výpočetně nemožné najít dvě různé zprávy M_1 a M_2 takové, že platí $h(M_1) = h(M_2)$.
- Lavinový efekt: pokud změníme 1 bit zprávy, signifikantně se změní výsledný hash (např. každý bit s pravděpodobností 50%).

Funkce SHA je prakticky náhrada za starší šifrovací algoritmy MDA4 a MDA5, protože ty přestaly být považovány za bezpečné, jelikož byla vyvrácena jejich bezkoliznost (u MDA4 byl nalezen způsob jak aktivně nalézt kolize [12], u MD5 byl nalezen způsob, jak vytvořit dva soubory se stejným hashem [5]). Ačkoliv u SHA-1 byl nalezen pouze teoretický útok (např. [26]), doporučuje se používat minimálně SHA-2.

2.5 Socialist Millionaires' protokol

U Socialist Millionaires' problem (SMP) [15] chtějí dva milionáři vědět, zda-li jsou oba dva stejně bohatí. Je to varianta na původní Millionaires' problem [25], kde chtěli dva milionáři vědět, zda a kdo z nich je bohatší, aniž by ukázali jakoukoliv jinou informaci o jejich bohatství.

V OTR místo porovnávání bohatství se porovnává sdílená informace mezi Alicí a Bobem. Tak se může Eva dozvědět pouze, jestli je Alice schopná uhodnout odpověď napoprvé. Jinak se protokol zastaví a Eva se nic nedozví. Eva se sice může pokusit o MITM útok tím, že pošle nezměněnou SMP informaci, ale OTR pro tento případ porovnává SHA-256 hash vytvořený ze session ID, fingerprintu obou stran a originálního tajemství mezi Alicí a Bobem. Tento útok tak nevyjde, protože Alice a Bob vytvoří jiný hash z fingerprintů. Pro potřebu OTR je také zredukovaný počet zpráv na jak málo bylo možno. Kompletní modifikovaný SMP lze nalézt v 3.2.1.

¹⁾Je trochu složitější se vyznat ve FIPS standardech pro SHA. Prvně byl vydán FIPS 180 se 160 bitovým hashem, označený jako SHA-0, v roce 1995 FIPS 180-1, který měl maximální délku zprávy a hash stejný, ale byl bezpečnější, označený SHA-1. FIPS 180-2 přidal 256, 384 a 512 bitový hash, označený SHA-2. FIPS 180-3 přidal 224 bitový hash. FIPS 180-4 zavedl možnost jiných parametrů (mimo jiné prodloužení maximální možné délky zprávy) pro hashe délky 224 a 256 bitů.

2.6 Man-in-the-middle útok

Je způsob útoku na komunikaci dvou stran, kdy se útočník (zde běžně pojmenovávaný Mallory) vmísí mezi účastníky komunikace tak, že účastníci jsou přesvědčeni, že mluví pouze spolu navzájem. V reálu však komunikace jde přes Malloryho. Takže komunikace místo očekávaného (Alice \rightleftharpoons Bob), běží (Alice \rightleftharpoons Mallory \rightleftharpoons Bob). Při jednoduché výměně klíčů může Mallory také úspěšně infikovat komunikaci. Zde si Alice bude žádat zprávu $M1$ o bobův klíč k_B aby mu mohla poslat tajnou zprávu $M2$

<i>Alice</i>	$M1, Sign_A \rightarrow$	<i>Mallory</i>		<i>Bob</i>
<i>Alice</i>		<i>Mallory</i>	$M1, Sign_A \rightarrow$	<i>Bob</i>
<i>Alice</i>		<i>Mallory</i>	$\leftarrow k_B$	<i>Bob</i>
<i>Alice</i>	$\leftarrow k_M$	<i>Mallory</i>		<i>Bob</i>
<i>Alice</i>	$g^{k_M}(M2) \rightarrow$	<i>Mallory</i>		<i>Bob</i>
<i>Alice</i>		<i>Mallory</i>	$g^{k_B}(M2') \rightarrow$	<i>Bob</i>

Jak je vidět, jak Alice, tak Bob, jsou přesvědčeni, že mluví spolu, avšak Mallory dokázal poslat Bobovi pozměněnou zprávu $M2'$. Aby si byli opravdu jisti, že nejsou oběti tohoto druhu útoku, musí se nasadit další techniky, např.:

- Nasazení tajných klíčů.
- Výměna klíčů bezpečným kanálem: např. při osobním setkání.
- Nezávislá certifikační infrastruktura: Provádí se ověření klíče srovnáním informace u věrohodné autority.
- Kvantová kryptografie: založena na fyzikální podstatě komunikačního kanálu.

Kapitola 3

Popis OTR

V této sekci probereme obě verze OTR protokolu, důvod, proč bylo nutno vytvořit nový.

Myšlenka na bezpečnou komunikaci byla popsána v "Off-the-Record Communication, or, Why Not To Use PGP" [6], kterou napsali Nikita Borisov a Ian Goldberg v roce 2004 (popis naleznete v sekci 3.1). Avšak jen o rok později pánové Di Raimondo, Genaro a Krawczyk poukázali na možnou slabinu této verze OTR [9], a to za použití „Diffie et al.'s identity misbinding“ útoku [11]. Proto v roce 2005 byla uvedena nová verze OTR [1], popsáno v sekci 3.2. Druhá verze navíc přinesla zjednodušení pro uživatele – už od něj není vyžadována jakákoliv znalost klíčů či fingerprintů.

3.1 OTR v.1

3.1.1 Šifrování

Zpráva, která má být přenesena se zašifruje stream šifrováním, zde použito AES, viz 2.3. Šifrovací klíč je určen pomocí Diffie-Hellman sdíleného tajemství, viz 2.1. Pro zajištění perfect forward secrecy se může Alice a Bob rozhodnout zahodit předchozí DH klíče x_A a x_B a stvořit nové. V tomto bodě se už nedá rekonstruovat předchozí zprávy, ani se znalostí g^{x_A} a g^{x_B} . DH algoritmus naštěstí není příliš výpočetně náročný, proto se dá měnit každých několik sekund i na nejslabších přístrojích. Aby se nemusely posílat zbytečně zprávy, připojí se nový klíč k posílané zprávě. Každá zpráva obsahuje DH veřejný klíč g^x podle kterého se odvodí klíč pro následující zprávu.

Výměna zpráv vypadá:

$$\begin{aligned} A &\rightarrow B : g^{x_1} \\ B &\rightarrow A : g^{y_1} \\ A &\rightarrow B : g^{x_2}, E(M_1, k_{11}) \\ B &\rightarrow A : g^{y_2}, E(M_2, k_{21}) \\ A &\rightarrow B : g^{x_3}, E(M_3, k_{22}) \end{aligned}$$

kde $k_{ij} = H(g^{x_i y_j})$ jako výsledek 128-bit hashovací funkce H , jako například SHA-1 na elementu \mathbb{Z}_p^* , a $E(M, k)$ udává šifrování v AES-CTR za použití klíče k . Zpráva je

zašifrována pomocí sdíleného tajemství získaného z posledního klíče zaslaného druhou stranou a klíčem, který byl druhé straně odeslán. Nepoužijeme přiložený klíč až do další zprávy. Například, v poslední zprávě tady nahoře, Alice dostala klíč g^{y_2} od Boba a poslední klíč, který ona odeslala je g^{x_2} , takže klíč k zašifrování další zprávy je $H(g^{x_2 y_2})$. V praxi se musí přenášet i ID klíče, aby obě strany měly jistotu, který k_{ij} se zrovna používá. Je to z důvodu, že při běžné komunikaci se stává, že si účastníci nevyměňují zprávy pravidelně (tzn. $A \rightarrow B, B \rightarrow A, A \rightarrow B, B \rightarrow A$, atd.) a ani protokolem to není vyžadováno.

3.1.2 Zapomínání klíčů

Aby se dosáhlo forward secrecy, musí se po úspěšné výměně klíčů zapomenout ty staré. Ideálně, jakmile Alice pošle klíč g^n Bobovi, ihned zapomene klíč $g^{x_{n-1}}$. Vzhledem k tomu, že komunikační protokoly jsou většinou asynchronní, je možné, že od Boba ještě může přijít zpráva zašifrovaná klíčem $g^{x_{n-1}}$, tudíž si Alice ten klíč musí pamatovat do doby, než od Boba dojde zpráva za použití g^{x_n} . Kdyby Alice posílala zprávy, aniž by Bob odpověděl a tvořila by při každé odeslané zprávě nové klíče $g^n \dots g^m$, musela by si celou posloupnost klíčů pamatovat, protože by si nemohla být jista, kterým klíčem Bob zašifruje svou zprávu. Proto (aniž by to mělo vliv na bezpečnost), Alice může vytvořit nový klíč až po Bobově reakci. Tím pádem si musí pamatovat pouze dva klíče – při příjmu zprávy, která použila g^{x_n} může zapomenout $g^{x_{n-1}}$, ale vytvoří si nový $g^{x_{n+1}}$ který bude použit v další zprávě. Samozřejmě, že pokud Bob delší dobu neodpoví, Alice může přečíst větší množství zpráv. Takže by bylo dobré, kdyby Bob čas od času poslal prázdnou zprávu, kvůli vygenerování nového klíče.

3.1.3 Autentizace

Pro autentizaci se používá MAC funkce, viz 2.2. MAC klíč vytvoříme, že použijeme hashovací funkci na dešifrovací klíč. To zapříčiní, že kdo je schopen přečíst zprávu, může i upravit MAC klíč. Což znamená, že i když Eva dokáže nějakým způsobem získat šifrovací klíč, nemůže dokázat, kdo tu zprávu poslal. Mohla to být Alice, Bob nebo i Eva sama.

Šifrovací klíč je výsledkem DH sdíleného tajemství, které ale potřebuje také být autentizováno. Toho se dá docílit digitálním podpisem DH výměny

$$\begin{aligned} A &\rightarrow B : \text{Sign}(g^{x_1}, k_A), K_A \\ B &\rightarrow A : \text{Sign}(g^{y_1}, k_B), K_B \end{aligned}$$

Kde k_A a K_A jsou Aliciny privátní a veřejné dlouhotrvající podpisové klíče a k_B , K_B jsou Bobovy. Pokud už Bob zná Alicin veřejný klíč, tak má jistotu, že g^{x_1} přišlo od Alice a že $g^{x_1 y_1}$ je známo jenom jim dvěma. A tak může zprávu autentizovanou klíčem $H(g^{x_1 y_1})$ opravdu považovat jako poslanou od Alice.

Zmíněný způsob autentizace je hybridní – používají se jak digitální podpisy i MAC. Digitální podpisy jsou dobré k tomu, že není potřeba držet $O(n^2)$ předpřipravených sdílených tajemství, ale jsou vytvářena pouze, když jsou potřeba. MAC autentizace zprávy nám zase zajistí repudiation.

Digitální podpis je potřeba pouze pro prvotní výměnu klíčů. Při dalších výměnách už používáme MAC k autentizaci klíčů, které používají staré sdílené tajemství. Pak zpráva vypadá:

$$g^{x_{i+1}}, E(M_r, k_{ij}), MAC(\{g^{x_{i+1}}, E(M_r, k_{ij})\}, H(k_{ij}))$$

Takže pokud víme, že prvotní autentizační klíč je bezpečně vyměněn, všechny následující budou také.

3.1.4 Odhalení MAC klíčů

Pro zvýšení bezpečnosti se udělá, na první pohled nesmyslný, krok – jakmile Alice pošle všechny zprávy Bobovi, které používají jeden MAC klíč, připojí ten klíč k další zprávě. Bob už do té doby zkontroloval všechny zprávy, které od Alice přišly. Nyní však může kdokoliv dodatečně napsat zprávu, která ten klíč používá a nikdo nemůže být bezpečně určen jako autor.

3.1.5 Útok na OTR v.1

Útok spočívá v tom, že Eva může vstoupit do inicialní výměny klíčů tak, že jak Alice, tak Bob dostávají stejné klíče, ale Alice věří, že mluví s Bobem, zatímco Bob věří, že mluví s Evou. Eva spustí MITM útok a začne konverzaci s Alicí a Bobem. Aliciny zprávy Bobovy Eva odchytí, a nahradí je identickými zprávami se svým podpisem. Zprávy od Boba Alici zůstanou stejné. Výměna zpráv tedy vypadá:

$$\begin{aligned} A &\rightarrow E : g^x, \text{Sign}_{s_A}(g^x), v_A \\ E &\rightarrow B : g^x, \text{Sign}_{s_E}(g^x), v_E \\ B &\rightarrow E : g^y, \text{Sign}_{s_B}(g^y), v_B \\ E &\rightarrow A : g^y, \text{Sign}_{s_B}(g^y), v_B \end{aligned}$$

Alice stále přijímá zprávy podepsané g^y a tak předpokládá, že mluví s Bobem. Na druhé straně, Bob přijímá g^x , podepsané Evou a předpokládá, že mluví s Evou, zatímco doopravdy mluví s Alicí.

Jako možné řešení této bezpečnostní díry bylo nasazení SIGMA protokolu [18]. Pak obě strany čekají, dokud se neurčí sdílené tajemství g^{xy} a teprve pak zašlou zprávu, která je identifikuje. Vzhledem k tomu, že Eva nezná to sdílené tajemství, tak nemůže pokračovat ve svém MITM útoku. Výměna zpráv pak vypadá:

$$\begin{aligned}
A &\rightarrow B : g^x \\
B &\rightarrow A : g^y \\
A &\rightarrow B : A, \text{Sign}_{s_A}(g^y, g^x), \text{MAC}_{K_m}(0, A), v_A \\
B &\rightarrow A : B, \text{Sign}_{s_B}(g^x, g^y), \text{MAC}_{K_m}(1, B), v_B
\end{aligned}$$

Zde, MAC klíč K_m je hashem g^{xy} , takže je neznámý Evě. Přidáním tohoto MAC je zabráněno identity misbinding útoku.

3.2 OTR v.2

3.2.1 Modifikovaný Socialist Millionaires' protokol

Pro tento protokol je potřeba mít další dva generátory g_2 a g_3 , oba jsou vytvořeny pomocí DH výměny. Alice si zvolí $a_2 \in \mathbb{Z}_q$ a Bob $b_2 \in \mathbb{Z}_q$. Pak si vymění $g_1^{a_2}$ a $g_1^{b_2}$ a spočítají si $g_2 = g_1^{a_2 b_2}$. Pak zopakují tento proces – zvolí si nové hodnoty a_3 a b_3 a vymění si $g_1^{a_3}$ a $g_1^{b_3}$, aby dostali $g_3 = g_1^{a_3 b_3}$. Nyní si Alice a Bob uloží hodnoty a_3 a b_3 použité při generování g_3 pro pozdější použití.

Pro skrytí x a y si Alice vybere $a \in_R \mathbb{Z}_q$ a spočítá $(P_a, Q_a) = (g_3^a, g_1^a g_2^x)$. Bob si obdobně vybere b a (P_b, Q_b) . Poté si oba vymění P_a, Q_a, P_b a Q_b .

Alice použije svou hodnotu a_3 a spočítá $R_a = \left(\frac{Q_a}{Q_b}\right)^{a_3}$. Bob obdobně spočítá $R_b = \left(\frac{Q_a}{Q_b}\right)^{b_3}$ a tyto hodnoty si vymění. Alice a Bob si nyní mohou spočítat $R_{ab} = R_a^{b_3} = R_b^{a_3}$. Teď obě strany vědí, že:

$$\begin{aligned}
R_{ab} &= \left(\frac{Q_a}{Q_b}\right)^{a_3 b_3} \\
&= (g_1^{a-b} g_2^{x-y})^{a_3 b_3} \\
&= g_3^{a-b} g_2^{(x-y) a_3 b_3} \\
&= \left(\frac{P_a}{P_b}\right) (g^{a_3 b_3})^{(x-y)}
\end{aligned}$$

Na to, aby zkontrolovali, zda $x = y$ potřebují pouze zkontrolovat, jestli $R_{ab} = \left(\frac{P_a}{P_b}\right)$. Jen poznámka: nikdo nezná hodnotu $g_2^{a_3 b_3}$, a je to náhodný generátor G . Proto, pokud $x \neq y$, $\left(R_{ab} \cdot \frac{P_b}{P_a}\right)$ bude nějaký náhodný element z G , a pokud $x = y$, bude to 1. A vzhledem k tomu, že náhodnost $\left(R_{ab} \cdot \frac{P_b}{P_a}\right)$ nezávisí na entropii x a y , tento protokol funguje dobře i když je entropie malá – mohou to být i obyčejná slova.

3.2.2 Šifrování

Největší změna v této verzi oproti předešlé je přepracované inicialní výměna autentizačních klíčů. Jak bylo navrženo, nyní se pro AKE používá SIGMA varianta. Navíc OTR implementovalo variantu, která skryje veřejné klíče před pasivním útočníkem (passive adversary). A kde se dřív posílaly veřejné klíče nezašifrované, nyní se posílají zašifrované pomocí DH sdíleného tajemství.

Prvně se založí neautentizovaný DH kanál a provede se autentizace uvnitř tohoto kanálu. Kanál je založen na sdíleném tajemství a je 64 bitový, což je málo na to, aby odolalo brute-force útoku. Výsledek je, že počáteční závazek (commitment) zajišťuje, že g^x nebude založeno na g^y .

AKE potom vypadá

Alice:

1. Vybere náhodné 128 bitové číslo r
2. Vybere náhodné 320 bitové číslo x
3. Pošle Bobovi $AES_r(g^x), SHA-256(g^x)$

Bob:

1. Vybere náhodné 320 bitové číslo y
2. Pošle Alici g^y

Alice

1. Spočítá $s = (g^y)^x$
2. Pošle Bobovi r

Bob

1. Odšifruje g^x za pomoci r
2. Zkontroluje, jestli g^x souhlasí s již přijatým $SHA-256(g^x)$
3. Spočítá $s = (g^x)^y$

Účel zmíněného r je kvůli splnění technických omezení – většina IM protokolů povoluje pouze nějakou maximální délku zprávy. Ideálně by Alice v první části, kroku 3, jednoduše poslala Bobovi $SHA-256(g^x)$ jako závazek a v druhé části, v kroku 2, by poslala g^x na otevření tohoto závazku. Takže místo poslání celého g^x , schováme hodnotu g^x zašifrováním pomocí r a pošleme jí v první zprávě. Ono r odhalíme až ve druhé zprávě. Nyní už je zavedeno sdílené tajemství s . Pro autentizaci Alice a Bob použijí $SHA-256$ hashe, vypočítané z s , s různými prefixy k vytvoření série MAC a AES klíčů. Tyto klíče jsou pak následně použity při výměně informací během AKE. Aliciny a Bobovy veřejné/soukromé klíče nazveme (v_A, s_A) , resp. (v_B, s_B) . Protokol pak pokračuje:

Alice:

1. Spočítá MAC klíče a_1, a_2, b_1, b_2 a AES klíče a_3, b_3

2. Určí $keyid_A$ – sériové číslo pro g^x
3. Spočítá $M_A = MAC_{a_1}(g^x, g^y, v_a, keyid_A)$
4. Spočítá $X_A = v_A, keyid_A, sign_{s_A}(M_A)$
5. Pošle Bobovi $AES_{a_3}(X_A), MAC_{a_2}(AES_{a_3}(X_A))$

Bob:

1. Spočítá MAC klíče a_1, a_2, b_1, b_2 a AES klíče a_3, b_3
2. Použije a_2 k ověření $MAC_{a_2}(AES_{a_3}(X_A))$
3. Použije a_3 k dešifrování $AES_{a_3}(X_A)$ a dostane tak $X_A = v_A, keyid_A, sign_{s_A}(M_A)$
4. Spočítá $M_A = MAC_{a_1}(g^x, g^y, v_a, keyid_A)$
5. Použije v_A k ověření $sign_{s_A}(M_A)$
6. Určí $keyid_B$ – sériové číslo pro g^y
7. Spočítá $M_B = MAC_{b_1}(g^y, g^x, v_B, keyid_B)$
8. Spočítá $X_B = v_B, keyid_B, sign_{s_B}(M_B)$
9. Pošle Alici $AES_{b_3}(X_B), MAC_{b_2}(AES_{b_3}(X_B))$

Alice:

1. Použije b_2 k ověření $MAC_{b_2}(AES_{b_3}(X_B))$
2. Použije b_3 k dešifrování $AES_{b_3}(X_B)$ a dostane tak $v_B, keyid_B, sign_{s_B}(M_B)$
3. Spočítá $M_B = MAC_{b_1}(g^y, g^x, v_B, keyid_B)$
4. Použije v_B k ověření $sign_{s_B}(M_B)$

Na konci tohoto protokolu, jak Alice, tak i Bob, vlastní sdílené tajemství s a vyměnili si key ID, takže mohou začít s tvořením nových klíčů, jak je popsáno dříve v 3.1.1. Alice také zná bobův veřejný klíč v_B a ví, že Bob zná svůj soukromý klíč s_B . Bob je na tom obdobně. Navíc, všechny tyto hodnoty byly zašifrovány a tak jsou v bezpečí před pasivním útočníkem.

Kapitola 4

Aktuální použití OTR

OTR team hned při zveřejnění tohoto protokolu předvedli i implementaci v IM komunikátoru. Vybrali si k tomu populární linuxový komunikátor GAIM¹⁾, který implementuje několik IM protokolů jako ICQ, AIM, jabber a další. Proto není OTR protokol závislý na konkrétním IM protokolu (např. jako na již zmíněném ICQ), ale na podpoře v rámci komunikátoru. Další vlastností tak je, že uživatel v rámci svého oblíbeného programu může komunikovat jak s přáteli přes nezabezpečený kanál, tak i zabezpečeně (samozřejmě, pouze pokud druhá strana má nainstalovanou potřebný plugin), protože jejich implementace automaticky zjistí, zda druhá strana je schopna komunikace přes OTR a pokud ano, tak ji nabídne k použití. Veškeré šifrování probíhá na straně klienta a přes samotný komunikační protokol probíhá komunikace jako běžné zprávy. Tudíž OTR protokol může být nasazen na jakýkoliv IM protokol.

4.1 IM komunikátory

Některé IM komunikátory nabízejí možnost OTR takzvaně „out of the box“ (tzn. už je v nich implementován). Mezi ně patří například Kopete a CenterIM. Pro jiné, jako např. již zmíněný Pidgin, Miranda, Trillian nebo Psi existují tzv. „third-party“ (stvořené někým externím) pluginy.

Pro IM komunikátory, které nemají přímou OTR podporu je ještě možnost použití proxy – k dispozici je SOCKS5, HTTP a HTTPS. Mezi ně patří např. AOL Instant Messenger a iChat.

4.2 Knihovny

K dispozici jsou i samotné knihovny:

- Knihovna libotr v jazyku C, vytvořena OTR teamem.
- Python binding pro libotr.

¹⁾Kvůli obsahu slova AOL v této zkratce se musel po nátlaku přejmenovat. Zvolili si nakonec Pidgin.

- Dvě knihovny v jazyce Java.
- Implementace v jazyku Scheme.

4.3 OTR přes SMS

Existuje i aplikace pro Android jménem TextSecure vyvíjena firmou Whisper Systems, která (mimo jiné) používá modifikovaný OTR protokol pro zasílání SMS. Modifikace spočívá v tom, že kvůli úspoře místa, místo DH výměny klíčů používá EEC klíče. Poté, co Whisper Systems byl koupen Twitterem, byl TextSecure uvolněn jako open-source.

II. PRAKTICKÁ ČÁST

Kontaktoval jsem jednoho z autorů tohoto protokolu (I. Goldberg) ohledně mé bakalářské práce. Navrhnul mi, že by bylo dobré implementovat tento protokol v jazyku JavaScript, který jsem si tedy vybral pro zpracování. Samotná aplikace je napsána v rozsahu potřebném k demonstraci mechanismů OTR protokolu a není tak zatím použitelná pro opravdové nasazení.

Výstup je formátován pomocí HTML s vloženými skripty. Pro psaní jsem používal PSPad a pro odlaďování vestavěnou JavaScript konzoli v prohlížeči Chrome.

Samotný projekt se skládá z:

- HTML soubor `otr.html` a soubor s CSS stylem `style.css`.
- Soubor s funkcemi `js/otr.js`
- Soubory JavaScriptu vytvořené třetími stranami v `js/bigint/` a `js/cryptojs/`, popsané v kapitole 5.

Má aplikace obsahuje kompletní prvotní výměnu klíčů, popsané v 3.2.2 a výměnu zašifrovaných zpráv. Oproti knihovně `libotr`, která je použita pro implementaci OTR do IM messengerů, v mojí aplikaci například chybí:

- Reakce na případné útoky na komunikaci, jako např. MITM útok.
- Socialist millionaires' protokol (ten se používá na vyžádání).
- Management klíčů pro asynchronní výměnu zpráv.

V sekci 6 je průvodce aplikací s komentáři.

Kapitola 5

Použitá řešení třetích stran

Pro vývoj mé aplikace jsem použil některá hotová řešení třetích stran. Tyto soubory v JavaScriptu budu nazývat knihovnami.

5.1 BigInt

Knihovnu BigInt [2] vytvořil Leemon Baird v roce 2000 a byla naposledy upravena v roce 2009. Zabývá se počítáním velkých čísel s vlastním datovým typem `bigInt` (big integer) založeným na ukládání čísel v poli (array). Je velkým přínosem, protože JavaScript má maximální hodnotu integeru 2^{53} , což v kryptografických implementacích, kde se počítá s čísly v řádech tisíců bitů (např. 1024 bitové šifrovací klíče), je silně nepostačující hodnota. Typ `float` se nedá zase použít, protože potřebujeme znát každý bit v celé délce čísla.

Tato knihovna má implementované základní i pokročilé výpočetní operace používané v kryptografii. Například v mé aplikaci používám `powMod(x, y, n)`, v případě počítání DH sdíleného tajemství $x^y \bmod p$. Jak vstup tak výstup je typu `bigInt`. Dále ještě tato knihovna obsahuje generátor náhodného prvočísla `randTruePrime(k)` o délce k bitů pomocí Maurerova algoritmu [19]. Ten používám pro generování RSA klíčů. Výstup se dá převést do jakékoliv báze od dvojkové do báze ≤ 95 .

5.2 CryptoJS

Tuto knihovnu napsal Jeff Mott v roce 2009, momentálně v letošní verzi 3 [20]. Zabývá implementací důležitých šifrovacích a hashovacích algoritmů, jako např. SHA, HMAC a AES, které využívám ve své aplikaci. Je velice dobře nastavitelná, nabízí mnoho alternativ – používám např. HMAC-SHA256 a AES v CTR módu. Dále se dá i v AES-CTR nastavit vlastní šifrovací klíč a inicializační vektor. Samozřejmostí je i převod hashe či dešifrované zprávy do různých kódování, jako např. Base64 nebo UTF-8.

Kapitola 6

Aplikace v JavaScript

V této sekci je ukázaná samotná aplikace. V první části se popíše, jak probíhá autentizace účastníků a výměna DH sdíleného tajemství. V Druhé části bude popsán „komunikátor“ pro odesílání a přijímání zpráv.

6.1 DH autentizace účastníků a výměna sdíleného tajemství

Alice:	
Vybere náhodné 128 bitové číslo r :	926880B6445628BDC5C521BD9665B0FA
Vybere náhodné 320 bitové číslo x :	9A6F76DEAB5BD910B30F68BA0BCC59E8E500588E1CDBAA21672E42791457F9CCF763E7B65EB93C45
Vypočítá AES_ r (g^x) (schová si g^x):	U2FsdGVkX18RMD4OcSGYa9mNxmJKOYM311phIHSMbooi103TznYRsg/9Y+nEcr1A LkPXuBe6iBy+XyH4jfHmPGohpYAHlsunfVbDBfOFNsztlkr8fOdKuDYLcxUSSeN3 PhC71bBy+U68KkA7t4zhodoyDsQ/f2kFIbSCI8f5eX30lb39Nus1jFlzOZX08SLn EB5XBkn1MUQKJ9IzqLyy97an+FK61iZ9a5XOQF5B7FumfP82ltzPfcrop3rCaJzd uw5M6yg89xf54wiCvRuNC7drjKxQCgqMSiC6xnt/fBcBEkYVuL/VcnIarX2532cr sGtuJSBZDpMoyQ6ti+5/4xxPFov+PDVgor6IpBIPcv4KvYotOxo9RE3TGOGZ73IQ iATow/YguQQ2cd73iHeqxzuR16mY+6FD1+e91ezhOYKMGdss9LmofKgxNdrCa9w1 tSjuaNAh5ImsI00Jmt/U2P+s8gkMEROcCdk33jhynVWnaZvbgqec4Hu0ZrOpFCwJp Y2QEZBwX3HMsORVeMiNn79eL0KBR0bqr0cyjegtRUYA=
Vypočítá SHA(g^x):	30fab2c2e4f4927b0549ccd0a71063de16cb1c6b36487f3ea59eb03a96421154

Obrázek 1. Alice, 1. část

Na obrázku 1 si Alice zvolí 2 náhodná čísla. Číslo r pro použití jako heslo pro AES a číslo x na základě kterého se se bude provádět DH autentizace. Na generování náhodných čísel je použit generátor z knihovny BigInt, který však využívá základní generátor z JavaScriptu. Kvalita těchto čísel proto nebude tak vysoká, aby se mohla uplatnit v opravdové kryptografii. Dále Alice vypočítá $g^x \bmod p$ ¹⁾. Hodnota generátoru g je 2 a hodnota p je dána na základě [17] jako 1536 bitové číslo. Dále ještě Alice toto číslo g^x zašifruje pomocí AES-CTR s heslem r a vypočítá k němu i hash za pomocí SHA-256. Tyto dvě hodnoty pošle Bobovi.

Bob:	
Vybere náhodné 320 bitové číslo y :	1300186E6B1D530A2071A34BAF9354BD3740ECCB8505B2D16D4A6D24D13818E2F3809C50BA978F
Pošle Alici g^y :	11E9566328DFA7ED1D2BE584C8377D55F2020580662DA5BF29F214C96AE064A200AB6050DEDA0EF9 28BE283580DB8C12F1036E5C01E8E46153AC9F0AECCE12238A22C7AEA0E35ECF5709A068B5D071EB 54F090042C5CB7B60154E868CA39710E3EA5B781D24C26ED767BE113226551C4A3CEAC58D2B2D6 41745113B349574B474971E329DC516E5E5FF9109ACA1936D7044EB7D891A00D4818DF321879F7FD 85804032A89F572B4F50F667017DE581DA11376760A1D06117C734F8D939DC7E

Obrázek 2. Bob, 1. část

¹⁾Jak již bylo dříve zmíněno, ve všech výskytech g^x , $(g^x)^y$ atd. budou myšleno $g^x \bmod p$

Na obrázku 2 si Bob obdobně vygeneruje náhodné číslo y , spočítá k němu g^y , které pošle Alici.

Alice:	
Spočítá $s=(g^y)^x$:	5E51313E523FA7E694F440F55A619C10C1E6BF557395F67A2C18AC56D69C32663DBDA7C387608048576CADCA7E27B2FFFAF276AE27E0FB88D0A8F878D36D8FDBDD57B2A495114C0280500DD88942001A337C325C632144A408AA35876A52576B324F741ED50D959973BF34FEB8B399150C3CF4A152D4F077CDCA0D77718AE70B3DBA4FC989293976E5514CABEB04FBF5EC4FAB54B44B5149ABA8DC8DA11D5A8D64F51F93FD5BB9B97CE58236E38601C2F7CDB55D1C98A6099003AE529B66A55C
Pošle Bobovi r :	926880B6445628BDC5C521BD9665B0FA

Obrázek 3. Alice, 2. část

Na obrázku 3 si Alice pomocí g^y , které má od Boba vypočítá sdílené tajemství $s = (g^y)^x$. Pošle Bobovi hodnotu r .

Bob:	
Odšifruje g^x za pomoci r :	3BD9E7EDFFA8DB45CA474751B6509943B33D294DAC6649E84E691C8FBEB968CBFEED7BE64B4BA6ACAFBA7E730406221EBF4350B8BA739AC08CA74ABBF279FA60B1F583188711E64B51870FD258D97D0237895CF3512AB7CDE752BD4BFB8B616A5178D130823028BEE53B8C528F8395D4E72CDF2426FA3CE3FEEEA55BCCD0EBEBF57A40F89AC55E075048E5859F1E3CBCF7A0600C490069FB25EFF4C2F2A22482423C8E5762FCD7E2669FABE309EB4B268EB5C57D885FAC1E6F8406D0FBB28B
Zkontroluje, zda g^x souhlasí s přijatým $\text{SHA}(g^x)$	30fab2c2e4f4927b0549ccd0a71063de16cb1c6b36487f3ea59eb03a96421154 true
Spočítá $s=(g^x)^y$:	5E51313E523FA7E694F440F55A619C10C1E6BF557395F67A2C18AC56D69C32663DBDA7C387608048576CADCA7E27B2FFFAF276AE27E0FB88D0A8F878D36D8FDBDD57B2A495114C0280500DD88942001A337C325C632144A408AA35876A52576B324F741ED50D959973BF34FEB8B399150C3CF4A152D4F077CDCA0D77718AE70B3DBA4FC989293976E5514CABEB04FBF5EC4FAB54B44B5149ABA8DC8DA11D5A8D64F51F93FD5BB9B97CE58236E38601C2F7CDB55D1C98A6099003AE529B66A55C

Obrázek 4. Bob, 2. část

Na obrázku 4 Bob, za pomoci přijatého hesla r , dešifruje zpět hodnotu g^x , spočítá její hash a porovná hodnotu hashe s hodnotou, kterou obdržel dříve od Alice (na obrázku je pod hodnotou hashe boolean hodnota „true“). Nyní si už Bob může vypočítat sdílené tajemství $s = (g^x)^y$, které bude shodné s tím, co dříve na obrázku 3 dostala Alice (protože $(g^x)^y = (g^y)^x$).

Kvůli zjednodušení jsem zde pro přenášení a dešifrování zašifrovaného textu použil přímý přístup do proměnné. Je to kvůli tomu, že výstup z AES šifrovací funkce z knihovny CryptoJS je typu object, který obsahuje všechny potřebné informace pro následné dešifrování.

Na obrázku 5 si Alice spočítá klíče pro HMAC a AES. Obdrží je tak, že použije dříve spočítané sdílené tajemství s s různými prefixy a použije je pro generování klíčů. V následujícím textu použiji označení $h(b)$, kde h je hashovací funkce SHA-256 a b je prefix ke sdílenému tajemství s , čili v kódu JS "b"+"s"

- Spočítá a_1 jako $h(2)$
- Spočítá a_2 jako $h(3)$
- Spočítá b_1 jako $h(4)$
- Spočítá b_2 jako $h(5)$
- Spočítá klíče a_3 jako prvních 128 bitů a b_3 jako následujících 128 bitů z $h(1)$

Alice:	
Spočítá MAC klíče	
Pro SHA-256-HMAC	a1: bc124e1507b79e77363e07dd993786ddaa333db6523ad42fb50b68b8c4e4a6eb a2: 21e624132ebb21d71f0a87c7ec7265d1216fa2a36c6f6182f5669e9f290ba9a3 b1: 5c4ffb8d8f3d466e07415f62e078eb5bf70c26e4fe1e6b5c1c150d8c7c3219ba b2: 77c30165149330724a8c712638e1acc51e366a71f7f1dd5d709eca3d860d9733
Pro AES	a3: 16addb937a757ba19389473f3dd994fe b3: a6e1f7de4951ce7b47da2154670a6729
Určí keyID_A pro g^x	1
Spočítá $M_A = MAC_{a_1}(g^x, g^y, v_a, keyid_A)$ ($g^x, g^y, v_a, keyid_A$)	dfc8c7dbe759309c4bbf7e4bfbcb78c9e0339b89f5c68ed58fb8a964a99246c2
Spočítá $X_A = v_A, keyid_A, sign_{s_A}(M_A)$ (M_A)	A2915D4D7F133AA28558847D3707C69981726304DA39117E8A5AE0FC3BAD2974866CDF91D984262 EE9D89EB3AEDE7F9CCBD6F1148EDEF3EF14594BE1CF0C09EADBEA0DBFC99346A79EB12E944E1FD4 42DEF4449616A0518A9E671581C72C663055606FCE005C04B5F1C9F88BD1AED7576884897C38B4D 8686764ED3776081, 10001 1 66B5A2ABB015C96024C55D3ADFF0CEE711E5B309A34E6CE9E1D23A4 4527E10ADB5C0A503658686627E2E4C66C4CD79D0ACD4489E64C200ABFD1124C428DAA7B77D387 DACE7CC6EC09932E56F7DAC92099F18FC916E01A5723E419B56AB7F11BEBB0481B2161F8B567BCD6 F01C70DDC63070A395092E5C7FD4DC3F47506E67E
Pošle Bobovi AES_{a_3} (X_A), MAC_{a_2}(AES_{a_3}(X_A))	U2FsdGVkX1+WNF/mCMVpXQFBo76Iibj18qrT2wdQbsWjaq3M7Q17cqnfcXqtGd 9uEkBMXJlrjpye/gilqmiGwj7sU4DaBDixfJpwU5lq5xS/lyI4pOaEPEKEZdjHza dgcj03VAcDUL/bQ7KjSHhB/pqegw5Eun0KyiQuBpV5dD8wJfPqzxpKEDP6EFN6 WjorEsn0p/and5gh/lSxKRGLx5swVBwEoksJwoKUzWJi fB8TbDEWfm2N1a2GCmD3 BEP7T8qbIX0nr5SFhuwpCkiRDx6buV3PrHpBPKVj2DA6gWkaSleIrSoCxmMjzL5P e+eOKu9an5aAhnvK4dP6KiXQsr6W6ooLNzDz18RgT5hI2iPIfo1+fD175dS3sY3y 3g0/wMDeDdvqL+a+DLz0l7ROV7pGLoiF+O/qxrIFxI0dr0pRS9BOWBER/xDs/EUW A/jUFjBW9IEQi5xulIU+uc2y86erPlwMBUC09hIT3UWC764ccihB6sL8jEaT76j2 gSILQtcn2Oi.fw68wLEBMP8qeBVN99pwHtv1z6xglQ7nkirR+zQbkL0c40G2OfyT nvwRGfry3524V2oJsuV1AEUFqbHcQXX0mcoxBoUbyY/ofBiUKZjov6mHKLvnlL47 bHGMZ+ChyDA9govL2tPdsTKqIAbK0VsbqCv0DB608rMB9q1PdpHhvhfPOEXSICTN 5zmo3bExzYH8wLz+v6LRWg== 7ad651009a0e8783666d0205da30e66ec00d580bb46f124350b7ee0 £726cc7ee

Obrázek 5. Alice 3. část

Určí hodnotu $keyid$ pro g^x , která musí být > 0 . Další její vlastností je, že později, při generování nových klíčů, u každého nového klíče vzroste o 1.

Spočítá $M_A = MAC_{a_1}(g^x, g^y, v_a, keyid_A)$ a $X_A = v_A, keyid_A, sign_{s_A}(M_A)$. Zašle Bobovi $AES_{a_3}(X_A), MAC_{a_2}(AES_{a_3}(X_A))$.

Pro podpis $sign_{s_A}(M_A)$ původní knihovna libotr používá algoritmus DSA. Tu jsem bohužel nenašel naimplementovanou v JS a tak jsem místo toho chtěl použít (již hotovou) knihovnu RSA, která mi bohužel nevyhovovala. Proto jsem si napsal vlastní jednoduchou implementaci podepisování podle RSA. V této aplikaci pro každé spuštění generuji nový pár soukromého/veřejného klíče.

Na obrázku 6 si Bob vypočítá klíče stejným způsobem jako Alice na obrázku 5. Jak je vidět, Alice i Bob došli ke stejným klíčům. Ihned si pomocí klíče a_2 ověří přijatý $MAC_{a_2}(AES_{a_3}(X_A))$ a pomocí klíče a_3 dešifruje $AES_{a_3}(X_A)$ a dostane tak X_A , která obsahuje mimo jiné alicin veřejný RSA klíč. Spočítá $M_A = MAC_{a_1}(g^x, g^y, v_a, keyid_A)$ a pomocí veřejného alicina klíče v_A ověří $sign_{s_A}(M_A)$. Zde na jsou na první pohled dvě rozdílné hodnoty, je to kvůli tomu, že knihovna BigInt používá v šestnáctkové soustavě ke značení hodnot vyšších než 9 velká písmena (ABCDEF), zatímco CryptoJS malá (abcdef). Proto, než je možno tyto dva řetězce srovnat `string==string`, musí se převést na stejný způsob zápisu (např. pomocí `string.toUpperCase()`). Dále ještě Bob spočítá $M_B = MAC_{b_1}(g^y, g^x, v_B, keyid_B)$ a $X_B = v_B, keyid_B, sign_{s_B}(M_B)$. Pak pošle Alici $AES_{b_3}(X_B), MAC_{b_2}(AES_{b_3}(X_B))$.

Na obrázku 7 Alice použije b_2 k ověření $MAC_{b_2}(AES_{b_3}(X_B))$, dešifruje $AES_{b_3}(X_B)$ pomocí b_3 a dostane tak X_B . Pak spočítá $M_B = MAC_{b_1}(g^y, g^x, v_B, keyid_B)$ a pomocí v_B ověří $sign_{s_B}(M_B)$.

Bob:	
Spočítá MAC klíče	
Pro SHA-256-HMAC	b_a1: bc124e1507b79e77363e07dd993786ddaa333db6523ad42fb50b68b8c4e4a6eb b_a2: 21e624132ebb21d71f0a87c7ec7265d1216fa2a36c6f6182f5669e9f290ba9a3 b_b1: 5c4ffb8d8f3d466e07415f62e078eb5b7f0c26e4fe1e6b5c1c150d8c7c3219ba b_b2: 77c30165149330724a8c712638e1acc51e366a71f7f1dd5d709eca38d60d9733
Pro AES	b_a3: 16addb937a757ba19389473f3dd994fe b_b3: a6e1f7de4951ce7b47da2154670a6729
Použije a_2 k ověření MAC_{a_2} (AES_{a_3}(X_A))	7ad651009a0e8783666d0205da30e66ec00d580bb46f124350b7ee0f726cc7ee true
Použije a_3 k dešifrování AES_{a_3}(X_A) a dostane tak X_A	A2915D4D7F133AA28558847D3707C69981726304DA39117E8A5AE0FC3BAD2974866CDF91D984262 EE9D89EB3AEDE7F9CCBD6F1148EDEF3EF14594BE1CF0C09EADBEA0DBFC99346A79EB12E944E1FD4 42DEF4449616A0518A9E671581C72C6630556060FCE005C04B5F1C9F88BD1AED7576884897C38B4D 8686764ED3776081, 10001 1 66B5A2ABB015C96024C55D3ADFF0CEE711E5B309A34E6CE9E1D23A4 4527E10ADB5C0A5036586086627E2E4C66C4CD79D00ACD4489E64C200ABFD1124C428DAA7B77D387 DACE7CC6EC09932E56F7DAC92099F18FC916E01A5723E419B56AB7F11EBBB0481B2161F8B567BC6E F01C70DDC63070A395092E5C7FD4DC3F47506E67E
Spočítá M_A=MAC_{a_1} (g^x, g^y, v_a, keyid_A)	dfc8c7dbe759309c4bbf7e4bfac78c9e0339b89f5c68ed58fb8a964a99246c2
A použije v_A k ověření sign_{s_A}(M_A)	DFC8C7DBE759309C4BBF7E4BFACB78C9E0339B89F5C68ED58FB8A964A99246C2 true
Uřídí keyid_B --sériové číslo pro g^y	1
Spočítá M_B=MAC_{b_1} (g^y, g^x, v_b, keyid_B)	f659f33306cf5195e3313de7f9b61f5022b1b39a846ff1c91968c574d55c5b13
Spočítá X_B=v_b, keyid_B, sign_{s_B} (M_B)	6025D76D59FCE513C74034345213BCFD25A5A1D26FDC27ADF64C9EF98F64D47BA1E24C467194237 62B62F857A685B6E174579106ECD2227CE697360F5A33ED6577C415ED054148A3B2DF15F5449BDBE 97DB60F9BF5BF0F0152c77a7924DFE1426CEE92E5DCD5A5EC0CB45FAF9D0ECAC6F040D60FDF0737 9A700CA8F394F637, 10001 1 3296B4B6DAA86A942A6ACB53A1D7933FC45686AFAEB9F1FECAB49E 8D6B93D33AE803B2C4BDE7CCEEB12F044AD140905EBA019D3C3AAC6256F6CE7949A83C96608BA803 B122964FB9F9CC126185D40508F02FBE33C10F86EECCDE309F3EFF5729ECF43C8051EE3486BCC7 4849C189B625640B94D4877964DC9EA0B6447C855
Pošle Alici AES_{b_3}(X_B), MAC_{b_2} (AES_{b_3}(X_B))	U2FsdGVkX1/YJL09R8oxqf5LJRmeFfXEOcOY7y4VDqfNOX/MFDOLWvdbwONbyCEE +LdKN3fcI/+W1gsfk7ws2RbX4fwwJx1H9o+Xegr03q6EGMI+StqppEgalLA+WxI4 7L7kPzefhCLQ4NV4gDT7kc712kwPvqV+k+PVDo0vBSe5ci21NX3AFj91lM/GaceXn mSohYrIk84kzIZhWvW3QyxJdfvDvDhWbcKMXQd4VDOJpQf6PuuHdU7svW7p8cEs0 23/1S2gr17VKg5GK5aigVpNJKCqXEMpQeQX3IhumMebUPj2snkgtv6RVMPQp9NA2 bLPX1tuA/IJPlXb00keJ5DLPC216c0NY6ayOtJSeQYzHE+RmC/w2Sm8W/NDtPuM 2PqG7SMKQePvWSyocKMeaTfx9GdbkoTx8zv0LpIJHUOD09L4o9fEH+eSv11y3syZ N2sReAi2Io+Gsc6C/iSYM12nH1Rhbc1TxuJNi5TyxzRCr6CTdV3Bc2Wbop6GqLV wFxdFjX2MfhckpSPSNu31a/9H1xi5e6ijHi9gc20vrUOYQLZtWnYwsXhdfvX6emN I5er/ovxNqeYusLgRixPqaPvPzthX7VouNY24gL+EW18NPLtrR5fgjQGi95qWmb I7jCDNZAO+symFdZma3JSJhKUuz+1EE1DSg/TwsPbuA20wEdK+I9AH6wZiPFOAR c56D6vGLv9aT/iz6/SUSQ== f0e1a5f1869e0ba370a754d34c6ff3a06fc599e85102b1b1b8a12a 0c53b3975

Obrázek 6. Bob, 3. část

Na konci této části jsou bezpečně vyměněny iniciální DH klíče, díky kterým jsme schopni šifrovat naši komunikaci a generovat nové klíče pro další zprávy. Díky RSA podepisování jsme si ověřili totožnost druhé strany.

6.2 Messenger

Jak Alice, tak Bob, má vlastní komunikační rozhraní. Hned na začátku jsou automaticky vyplněny známé hodnoty „Tajné DH x“ a „Nový přijatý DH (g^y)“ (tzn. hodnoty, které se vygenerovaly a vyměnili v předchozích autentizačních zprávách). Jak vypadá iniciální stav je na obrázku 1. Jediné pole, které se dá upravovat je u tlačítka „Send message.“

Po napsání zprávy a stisknutí tohoto tlačítka, se zpráva odešle Bobovi a objeví se mu v poli vedle tlačítka „Receive message“. Zároveň se u Alice objeví „Nový alicin MAC klíč“. Struktura zprávy je následující:

- g^{x+1} (tzn. klíč pro další zprávu).
- Zašifrovaná zpráva (ve formátu čitelném pro AES z knihovny CryptoJS).
- MAC zprávy – zde MAC vypočítaný z g^{x+1} a zašifrované zprávy.

Alice:	
Použije b_2 k ověření MAC_{b_2} (AES_{b_3}(X_B))	f0e1a5f1869e0ba370a754d34cf6ff3a06fc599e85102b1bab8a12a0c53b3975 true
Použije b_3 k dešifrování AES_{b_3}(X_B) a dostane tak X_B	6025d76d59f513c74034345213bcfd25a5a1d26fDC27ADF64C9EF98F64D4A7BA1E24C46719423762B62F857A685B6E174579106ECD2227CE697360F5A33ED6577C415ED054148A3B2DF15F5449BDBE97DB60F9BF5BF0F0152C77A7924DFE1426CEE92E5DCD5A5EC0CB45FAF9D20ECAC6F040D60PDF07379A700CA8F394F637, 10001 1 3296B4B6DAA86A9642A6ACB53A1D7933FC45686AFAEB9F1FECAB49E8D6B93D33AE803B2C4BDE7CCEEB12F044AD140905EBA019D3C3AAC6256F6CE7949A83C96608BA803B122964FB9F9CC126185D40508F02FBE33C10F86EECCDE309F3EFF5729ECF43C8051EE34866BCC74849C189B625640B94D4877964DC9EA0B6447C855
Spočítá M_B=MAC_{b_1}(g^y.g^x.v_B.keyid_B)	f659f33306c5f195e3313de7f9b61f5022b1b39a846ff1c91968c574d55c5b13
Použije v_B k ověření sign_{s_B}(M_B)	F659F33306CF5195E3313DE7F9B61F5022B1B39A846FF1C91968C574D55C5B13 true

Obrázek 7. Alice, 4. část

Alice:	
Přijatá zpráva:	
Tajné DH x:	9A6F76DEAB5BD910B30F68BA0BCC59E8E500588E1CDBAA21672E42791457F9CC763E7B65EB93C45
Nový přijatý DH (g^y):	11E9566328DFA7ED1D2BE584C8377D55F2020580662DA5BF29F214C96AE064A200AB6050DEDA0EF928BE283580DB8C12F1036E5C01E8E46153AC9F0AECCE12238A22C7AEA0E35ECF5709A068B5D071EB54F090042C5CB7B60154E868CAA39710E3EASB781D224C26ED767BE113226551C4A3CEAC58D2B2641745113B349574B474971E329DC516E5E5FF9109ACR1936D7044EB7D891A00D4818DF3218797FD85804032A89F572B4F50F667017DE581DA11376760A1D06117C734F8D939DC7E
Ciphertext:	
AES IV a Salt:	
MAC:	
Starý alicin MAC klíč:	
Nový alicin MAC klíč:	
Starý bobův MAC klíč:	
<input type="button" value="Send message"/>	Nazdar Bobe!
<input type="button" value="Receive message"/>	

Obrázek 1. Iničiální stav messengeru s napsanou zprávou

- Starý MAC klíč – klíč, který byl použit při tvoření MAC minulé zprávy.

Oproti originálnímu protokolu zde chybí:

- Verze protokolu.
- Typ zprávy – např. SMP zpráva má jiný identifikátor.
- Flagy – jediný implementovaný je flag pro ignorování zpráv, které nejsou dešifrovatelné nebo ověřitelné.
- Odesílatelovo a příjemcovo keyid.
- Infomace o stavu counter kvůli CTR pro zprávy, které mají stejné (odesílatelův a příjemcův) keyid pár.

Po výměně několika zpráv vypadá alicin messenger jako na obrázku 2. Zprávy musí být v této aplikaci postupně (tzn. $A \rightarrow B, B \rightarrow A, A \rightarrow B$ atd.), jinak se nepovede

Alice:	
Přijatá zpráva:	Jo, fajn.
Tajné DH x:	984237D907C40B59BCFF20104614DF7BA9F73502975D4F951E086EBC3D5189FD73E9880A19E81A0F
Nový přijatý DH (g^y):	5DE56BE65AD4551CCF02CBEF7897C839ED10014A48A05232E52FD1E276CD889EA40CB181B9B55106FC EA644803FF3EA8C16ABF7AE41A325D740121045F948534A0AB37CB6D639E302A03A40ABBB3BAB7E144 BD73928EE7E3E9B37D602A3DA9E1FA60DFCADD0BDD92EDD8487BFC78A7AB4995727E50A3FAA5F7A456 020DB9EB8C5E215C8248D97E1D69CC073306020945F54929EE55191648C5301C97A7FDEBAA10187A0 1BD063B6F95687FE4B5611E804D294669BDADFA627CB00CAB9FC93F7
Ciphertext:	TZLZUC+8x5zJRL+wKy+iQA==
AES IV a Salt:	44c665bb5419456b4f37b7f8fcd83d78, 09b6417f7a126839
MAC:	9282eb0c365f42e5fbee8869d7fbc4d523fc96d3dbe9d785f3b197e62ab9a66c
Starý alicin MAC klíč:	13e5aff87e7edf6692ef2649c5660d73c5fffb517efc41e2296189ee2ef866b91
Nový alicin MAC klíč:	e10095475dbbc347d1d2091a5861c95a9004d3a0077d0e50c28b2923260a1fdb
Starý bobův MAC klíč:	ada6cacb45b87d87e9db4d115dee7f4f6faa4c8943d9b22ffd7576d13b1bc77d
<input type="button" value="Send message"/>	Jak se vede?
<input type="button" value="Receive message"/>	5DE56BE65AD4551CCF02CBEF7897C839ED10014A48A05232E52FD1E276CD889EA40CB181B9B55106FC EA644803FF3EA8C16ABF7AE41A325D740121045F948534A0AB37CB6D639E302A03A40ABBB3BAB7E144 BD73928EE7E3E9B37D602A3DA9E1FA60DFCADD0BDD92EDD8487BFC78A7AB4995727E50A3FAA5F7A456 020DB9EB8C5E215C8248D97E1D69CC073306020945F54929EE55191648C5301C97A7FDEBAA10187A0 1BD063B6F95687FE4B5611E804D294669BDADFA627CB00CAB9FC93F7 { "ct": "TZLZUC+8x5zJRL+wKy+iQA==", "iv": "44c665bb5419456b4f37b7f8fcd83d78", "s": "09b6 417f7a126839" } 9282eb0c365f42e5fbee8869d7fbc4d523fc96d3dbe9d785f3b197e62ab9a66c ad a6cacb45b87d87e9db4d115dee7f4f6faa4c8943d9b22ffd7576d13b1bc77d

Obrázek 2. Stav messengeru po několika zprávách

dešifrovat přicházející zprávy. Je to kvůli již zmíněnému chybějícímu managementu klíčů.

Obsah buněk je následující:

- Přijatá zpráva – poslední zpráva přijata od Boba.
- Tajné DH x – náhodné x na základě kterého bude šifrována další zpráva. Nové se vytvoří v průběhu posílání další zprávy Bobovi.
- Nový přijatý DH (g^y) – nové g^y vytvořené Bobem a posláno v poslední zprávě od něj.
- Ciphertext – zašifrovaná zpráva.
- AES IV a Salt – informace potřebné pro dešifrování textu pomocí AES (kvůli CryptoJS knihovně).
- MAC – MAC poslední přijaté zprávy.
- Starý alicin MAC klíč – klíč, kterým Alice podepsala MAC předposlední poslané zprávy. Byl dle myšlenky v sekci 3.1.4 přibalen v poslední poslané zprávě.
- Nový alicin MAC klíč – klíč, kterým Alice podepsala MAC poslední poslané zprávy.
- Starý bobův klíč – klíč, který byl přibalen v poslední přijaté zprávě, pomocí kterého byla podepsána předposlední přijatá zpráva.

- Send message – text poslední odeslané zprávy.
- Receive message – text poslední přijaté zprávy v kompletní formě.

Díky neustálé změně DH klíčů je dosaženo forward secrecy - jakmile jednou klíč zapomeneme už nejsme schopni rozluštit staré zprávy. Navíc při příjmu MAC klíče nastává situace, kdy tu zprávu mohl podepsat kdokoliv, tudíž jsme prokázali neprokatelnost autorství zprávy.

ZÁVĚR

Ačkoliv je spousta možností, jak zabezpečit svojí komunikaci, pouze Off-the-record protokol nám zajistí šifrovanou komunikaci, autentizaci účastníků, bezpečnost historie naší komunikace a neprůkaznost autorství zprávy.

V první části byl průřez historií šifrování od jejich prvopočátků až po moderní kvantovou kryptografii. Také byla vysvětleny základní rozdíly mezi steganografií a kryptografií nebo symetrickým a asymetrickým šifrováním.

V další části bylo blíže vysvětleny prostředky a jejich případné varianty, které jsou důležité pro OTR protokol. Navíc se vysvětlil princip Man-In-The-Middle útoku, který bylo možno aplikovat na první verzi OTR protokolu.

Následoval vlastní popis první verze OTR protokolu, její jednotlivé části a jejich přičinění k dosažení kýžených vlastností. Byl popsán i princip útoku na první verzi tohoto protokolu. Na základě tohoto zjištění byl přidány další prostředky do další verze OTR protokolu k opětovné zajištění bezpečnosti.

Byl i zjištěn aktuální rozšíření tohoto způsobu komunikace, že je naimplementován převážně na IM komunikaci vedenou přes Internet.

V praktické části jsme si pomocí aplikace v JavaScriptu demonstrovali způsob výměny šifrovacích klíčů, autentizaci účastníků, generování nových klíčů a odhalování starých klíčů. Díky těmto praktikám je možno dosáhnout všech požadovaných vlastností tohoto protokolu.

CONCLUSIONS

Although, we have many possibilities, how to protect our communication, only Off-the-record protocol is able to provide encrypted communication, participant authentication, perfect forward secrecy and repudiability.

In first part of was brief view of history of cryptography – from first occurrences to modern quantum cryptography. There was mentioned difference between cryptography and steganography as well as difference between symmetric and asymmetric ciphering. In next part was taken closer view to instruments which are used by OTR protocol and their modifications. Then was explained principles of Man-In-The-Middle attack which was possible to apply to first version of OTR protocol.

Next part was about description of OTR protocol, it's parts and their contribution for achievement of desirable attributes. Then was described principle of attack to first version of OTR protocol followed by needed actions for achieving security.

Then was find out situation of usage of OTR protocol and the conclusion that it's used mainly for IM communication over Internet.

In practical part was demonstrated with help of JavaScript application the way of ciphering key exchange, participant authentication, generating of new keys and uncovering old keys. Thanks to this practices is possible to achieve all desired attributes of this protocol.

Literatura

- [1] Alexander Ch. a I. Goldberg. Improved User Authentication in Off-The-Record Messaging. In Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society. ACM Press, 2007, s. 41–47.
- [2] BAIRD, L. BigInt. [online]. 2000, 2009 [cit. 2012-06-11]. Dostupné z: <http://www.leemon.com/crypto/BigInt.js>
- [3] BELLARE M., R. CANETTI a H. KRAWCZYK. HMAC: Keyed-hashing for message authentication, RFC2104. 1997. [online]. [cit. 2012-06-11]. Dostupné z: <http://www.ietf.org/rfc/rfc2104.txt>.
- [4] BÍLKOVÁ G.. Moderní kryptologie. 2010. Bakalářská práce. UTB ve Zlíně.
- [5] BLACK J., M. COCHRAN M. a T. HIGHLAND. A Study of the MD5 Attacks: Insights and Improvements. 2006. [online]. [cit. 2012-06-11]. Dostupné z: <http://www.cs.colorado.edu/~jrblack/papers/md5e-full.pdf>
- [6] BORISOV N., I. GOLDBERG a E. BREWER. Off-the-Record Communication, or, Why Not To Use PGP. In Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society. ACM Press, 2004, s. 77–84.
- [7] DAEMEN J. a V. RIJMEN. AES Proposal: Rijndael, AES Algorithm Submission, 1999.
- [8] DAEMEN J. a V. RIJMEN. The block cipher Rijndael, Smart Card research and Applications. LNCS 1820, Springer-Verlag, 2000, s. 288–296.
- [9] DI RAIMONDO M., R. GENNARO a H. KRAWCZYK. Secure Off-the-Record Messaging. In Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, ACM Press, 2005, s. 81—89
- [10] DIFFIE W. a M. E. HELLMAN. New Directions in Cryptography. In IEEE Transactions on Information Theory, vol. IT-22, 1976, s. 644–654.
- [11] DIFFIE W., P. C. VAN OORSCHOT a M. J. WIENER. Authentication and authenticated key exchanges. In Designs, Codes and Cryptography, 1992, s. 107–125.

- [12] DOBBERTIN H.. Cryptanalysis of MD4. In Journal of Cryptology, International Association for Cryptologic Research, 1998, s. 253–271.
- [13] DWORKIN M.. Recommendation for block cipher modes of operation: Methods and techniques. NIST Special Publication 800-38A, 2001.
- [14] HOUSLEY R.. Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP). RFC3686, 2004. [online]. [cit. 2012-06-11]. Dostupné z: <http://tools.ietf.org/html/rfc3686>
- [15] JAKOBSSON M.a M. YUNG. Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers. In Proc. of Advances in Cryptology—CRYPTO '96. Lecture Notes in Computer Science 1109, 1996, s. 186–200.
- [16] JOHNSON N. F.. Steganography. Technical report. 1995. [online]. [cit. 2012-06-11]. Dostupné z: http://www.jjtc.com/pub/tr_95_11_nfj/index.html
- [17] KIVINEN T. a M. KOJO. More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). RFC 3526, 2003. [online]. [cit. 2012-06-11]. Dostupné z: <http://www.ietf.org/rfc/rfc3526.txt>
- [18] KRAWCZYK H.. SIGMA: The ‘SIGn-and-MAc’ Approach to Authenticated Diffie-Hellman and Its Use in IKE Protocols. In Proceedings of CRYPTO '03, 2003, s. 400–425.
- [19] MAURER U. M.. Fast generation of prime numbers and secure public-key cryptographic parameters. In Journal of Cryptology: the journal of the International Association for Cryptologic Research, 1995, s. 123–155.
- [20] MOTT, J.. CryptoJS. [online]. 2009, 2012 [cit. 2012-06-11]. Dostupné z: <http://code.google.com/p/crypto-js/>
- [21] National Institute of Standards and Technology. Announcing the advanced encryption standard (AES). Federal Information Processing Standards Publication 197, 2001.
- [22] National Institute of Standards and Technology. Announcing the secure hash standard. Federal Information Processing Standards Publication 180-4, 2012.
- [23] PASEKA J.. Kryptografie. 2012. [online]. [cit. 2012-06-11]. Dostupné z: <http://is.muni.cz/el/1431/jaro2012/M0170/um/um/Finalkrypto2012.pdf>
- [24] SINGH S.. Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii. Praha: Dokořán, 2003, 382 s. ISBN 80-865-6918-7.

-
- [25] YAO A.. Protocols for secure computations. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 1982, s. 160–164.
- [26] WANG X., Y. L. YIN a H. YU. Finding collisions in the full SHA-1. In proceedings of CRYPTO '05, 2005, s. 17–36.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES	Advanced Encryption Standard
AES-CTR	AES counter mode
AKE	Authenticated Key Exchange
CSS	Cascading Style Sheets
CTR	Counter mode
DES	Data Encryption Standard
DH	Diffie-Hellman
EEC	Elliptic curve cryptography
FIPS	Federal Information Processing Standard
g	generátor grupy (prvočíslo)
GAIM	GTK+ AOL Instant Messenger
h	Hashovací funkce
IM	Instant Messaging
IV	Inicializační vektor (Initialization Vector)
JS	Javaskript
MAC	Message Authentication Code (kryptografický zkušební součet)
MD4/5	Message-Digest Algorithm 4/5
MITM	Man-in-the-middle
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OTR	Off-The-Record
p	Prvočíslo
PGP	Pretty Good Privacy
RGB	Red Green Blue (primární barvy v počítačové grafice)
S/MIME	Secure/Multipurpose Internet Mail Extensions
SHA	Secure Hash Algorithm
SMS	Short Message Service

Seznam obrázků

Obr. 1. Alice, 1. část.....	34
Obr. 2. Bob, 1. část	34
Obr. 3. Alice, 2. část.....	35
Obr. 4. Bob, 2. část	35
Obr. 5. Alice 3. část.....	36
Obr. 6. Bob, 3. část	37
Obr. 7. Alice, 4. část.....	38
Obr. 1. Iniciální stav messengeru s napsanou zprávou	38
Obr. 2. Stav messengeru po několika zprávách	39

SEZNAM PŘÍLOH

P I. CD

PŘÍLOHA P I. CD

bc-prace-jiri-vavra.pdf	Vypracovaná bakalářská práce
bc-prace-jiri-vavra.zip	Archiv se zdrojovým kódem vypracované práce v LaTeX
prakticka-cast.zip	Zdrojové kódy aplikace včetně knihoven třetích stran