BabelApp

Firemní platforma pro bezpečnou komunikaci

White Paper

Obsah

1. SPECIFIKACE PRODUKTU

BabelApp je platforma pro bezpečnou komunikaci. BabelApp umožňuje šifrovat hlasové hovory, zasílat a bezpečně ukládat zašifrované zprávy a přiložené dokumenty. Šifrování se provádí na koncových zařízeních, žádný server, správce, provozovatel, ani vnitřní nebo vnější útočník nemá přístup k rozšifrovaným datům. BabelApp používá silné kryptografické algoritmy a protokoly založené na mezinárodních standardech.Platforma BabelApp se skládá ze serverové části a klientů pro mobilní zařízení se systémy Android a iOS a počítače s operačními systémy Windows a macOS (OS X).

1.1 HLAVNÍ RYSY

Platforma BabelApp je založena na podporovaných koncových zařízeních (klientech) a serverech, které bezpečně komunikují s klienty i mezi sebou prostřednictvím internetu. Nepoužívají se žádné digitální certifikáty pro uživatele a jejich zařízení.

1.1.1 PODPOROVANÁ ZAŘÍZENÍ

Klienti BabelApp jsou k dispozici zdarma pro všechny významné platformy mobilních i stolních zařízení.

- klient pro iOS
- klient pro Android
- klient pro BlackBerry
- klient pro PC s Windows
- klient pro Mac s macOS

Každý uživatel může mít pod svým účtem registrováno více zařízení s libovolnou kombinací.
Každé zařízení může být připojeno k více serverům.

1.1.2 HLAVNÍ SLUŽBY SERVERŮ PRO KOMUNIKACI

BabelApp poskytuje prostřednictví serveru centrální komunikační služby:

- BabelApp signalizace pro VoIP
- STUN a TURN server pro VoIP
- Centrální adresář kontaktů
- Distribuce a synchronizace veřejných klíčů uživatelů
- komunikace mezi servery pro získání veřejných klíčů příjemce z cílového serveru a doručení zprávy uživateli z jiné domény
- asynchronní doručení zpráv a příloh na více zařízení příjemce
- synchronizace zpráv odeslaných z jednoho zařízení na ostatní zařízení odesilatele
- dočasné uložení zašifrovaných příloh pro stažení příjemcem zprávy
- brána pro zasílání požadavků na rozeslání Push notifikací o zprávách čekajících na doručení
- zasílání informací o doručení a přečtení zpráv nebo o nedoručitelnosti
- komunikační brána s REST API pro snadnou integraci s aplikacemi a programovatelnými zařízeními k rozesílání šifrovaných zpráv a automatizaci firemních procesů

1.1.3 FUNKCE SERVERŮ PRO SPRÁVCE

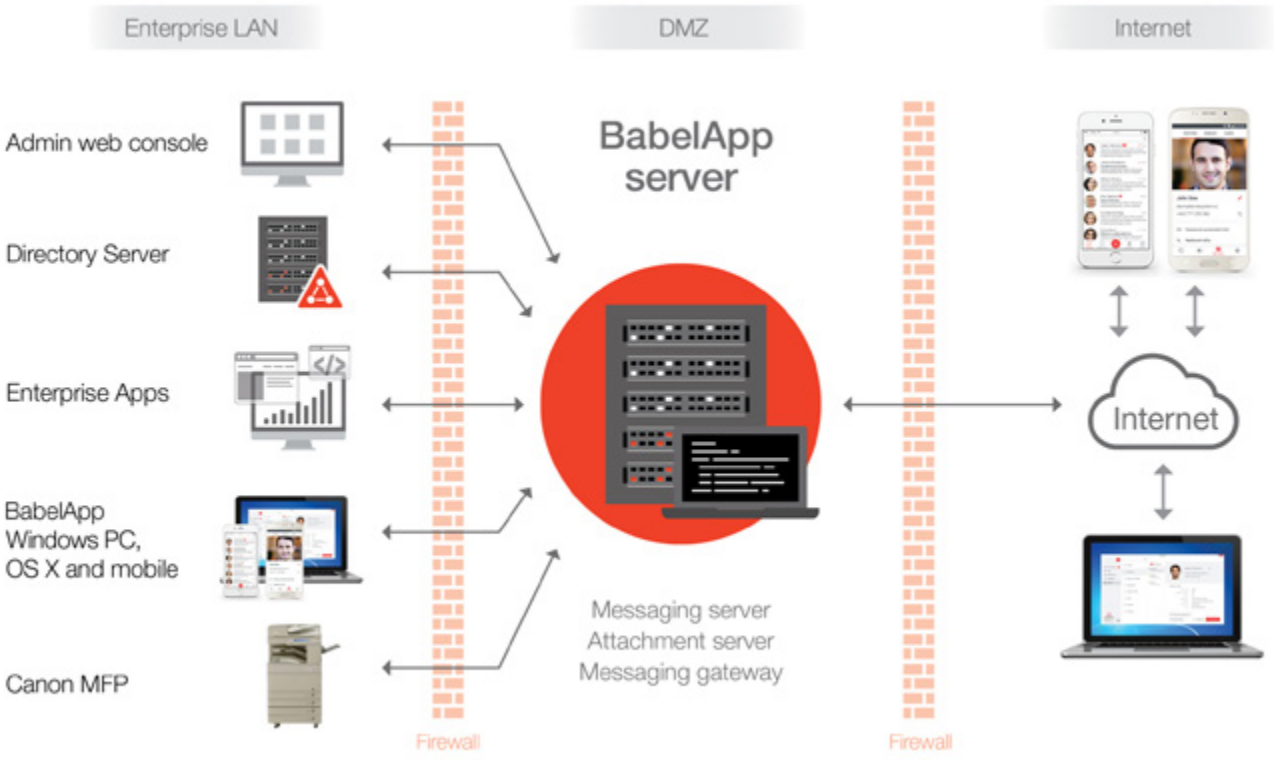
BabelApp dává plnou kontrolu nad správou a během celé infrastruktury do rukou firemního správce.

- web konzole pro správu systému
- import uživatelů z LDAP/AD
- synchronizace změn účtů z LDAP/AD
- možnost vytváření a správy uživatelských účtů a skupin pro regulérní členy a externisty
- registrace zařízení uživatelů pomocí OTP nebo LDAP autentizace
- odebrání registrovaného zařízení
- zneplatnění klíčů
- zablokování uživatele
- reset účtu serveru na vzdáleném serveru
- smazání vzdáleného serveru z databáze účtů lokálního serveru
- rozesílání zpráv ze serveru podle plánu
- systémové logování událostí serveru a uživatelů
- statistiky provozu

2. ARCHITEKTURA SYSTÉMU

BabelApp poskytuje robustní firemní platformu pro šifrovanou komunikaci mezi mobilními zařízeními a pracovními stanicemi, se snadnou integrací firemních aplikací a multifunkčních tiskáren/skenerů. Aby bylo možné komunikovat i s příjemci, kteří mají účty na jiných serverech, než odesilatel, implementuje BabelApp komunikaci mezi servery.

Pro dosažení vysoké dostupnosti podporuje server implementaci klastru odolného proti výpadku.



2.1 KOMPONENTY SERVERU

BabelApp server je tvořen sadou funkčních komponent:

- služba pro asynchronní zasílání zpráv
- služba pro stažení příloh
- centrální adresář kontaktů
- komunikační brána s API
- brána pro zasílání upozornění
- plánovač pro rozesílání zpráv serverem
- web stránky administrátora
- web stránky uživatele

Pro podporu uvedených služeb server využívá prostřednictvím konektorů služby firemního síťového adresáře:

- LDAP konektor
- SSO konektor

2.1.1 ASYNCHRONNÍ ZASÍLÁNÍ ZPRÁV: MESSAGING SERVICE

Základní službou serveru je podpora pro asynchronní zasílání zpráv založených na XMPP protokolu. Zprávy jsou ve formátu JSON, který obsahuje veškerá metadata nutná pro procesy doručení zprávy, kontrolu integrity, transport zašifrovaných klíčů zprávy, matadat o přílohách atd.

Zpráva může být určena pro více příjemců, může mít více částí, u každé části je informace o verzi protokolu, pro kterou je zpráva určena. Server zjistí aktuální adresy příjemců a verze protokolů podporovaných zařízeními příjemců a doručí jim nejvhodnější část zprávy se stejnou, nebo nejbližší nižší verzi protokolu.

Server přijme asynchronní zprávu, potvrdí odesílajícímu zařízení úspěšný příjem a uchová jí do chvíle, dokud nepřijme potvrzení, že zpráva je doručená, nebo dokud neuplyne maximální doba pro doručení; poté server zprávu smaže.

Příjemce může mít registrováno více zařízení, server se snaží doručit zprávu na všechna zařízení příjemce. Zpráva se považuje za doručenou, pokud byla doručena alespoň na jedno zařízení příjemce.

Pokud příjemce zprávu rozšifruje, server zašle odesilateli zprávu o přečtení.

Pokud se zprávu nepodaří doručit, zašle server odesilateli zprávu o nedoručitelnosti. Pokud byla zpráva určená více příjemcům, obsahuje zpráva o nedoručitelnosti adresy, na které nebylo doručeno.

Odesilatel může mít registrováno více zařízení, zprávu odeslanou z libovolného zařízení se server snaží synchronizovat na ostatní zařízení odesilatele..

2.1.2 SLUŽBA PRO STAŽENÍ PŘÍLOH: ATTACHMENT SERVICE

Možnost zasílání dokumentů ve formě příloh zprávy je volitelná, správce jí může povolit, nebo zakázat. Zprávy obsahují pouze metadata příloh, vlastní přílohy je nutné samostatně stáhnout na základě informací uvedených ve zprávě. Přílohy jsou šifrovány obdobně jako samotné zprávy, podrobně viz článek 3.3.6. Správce může nastavit omezení na maximální velikost přílohy a maximální dobu, po kterou jsou přílohy umístěny na serveru pro stažení; po překročení této doby jsou automaticky smazány.

2.1.3 BABELAPP VOIP SIGNALIZACE

Pro navázání telefonního spojení mezi klienty je nutná výměna zpráv prostřednictvím serveru (viz 2.1.1) v rámci signalizačního protokolu BabelApp. Signalizační zprávy jsou zasílány standardním transportním mechanismem pro zasílání šifrovaných zpráv s kontrolou integrity, viz 3.5.6 a 3.5.7. V rámci signalizace dojde k výměně veřejné části dočasných DH klíčů volajícího a volaného klienta a následně k dohodě na dočasném sdíleném tajemství, ze kterého jsou odvozeny jednorázové kryptografické klíče pro navazovanou SRTP relaci.

2.1.4 STUN SERVER

Mobilní zařízení a počítače jsou připojeny do internetu prostřednictvím směrovače (routeru), který provádí překlad adres (NAT) mezi privátními adresami ve vnitřní síti a veřejnými adresami v internetu. Pro zasílání BabelApp zpráv prostřednictvím serveru funguje NAT transparentně a bez problémů, pro přímou komunikaci mezi zařízeními při telefonování v rámci VoIP je NAT překážkou. NAT se skládá z několika mechanismů, jejichž kombinace určuje celkové chování NAT:

- mapování adres (NAT binding)
- mapování portů (Port binding)
- doba života relace a mapování (Binding timer)
- filtrování (NAT Filtering)

Pro přímou komunikaci mezi BabelApp klienty v rámci VoIP musí klient v lokální síti vědět, jak jeho adresa „vypadá z venku“, tj. jakým způsobem je přeložena NAT a jaký je typ NAT a tuto informaci musí poskytnout protistraně v rámci signalizačního protokolu BabelApp.

Mechanismus pro zjištění, zdali mezi komunikujícími stranami je jeden nebo více NAT a jak se příslušný NAT chová je specifikován v RFC 5389, Session Traversal Utilities for NAT (STUN).

2.1.5 TURN SERVER

TURN definuje protokol rozšiřující STUN, který umožňuje klientům za NAT požádat TURN server, aby přeposílal pakety mezi klientem A a klientem B. K tomuto účelu TURN server alokuje na základě požadavku klienta A „Relayed Transport Address“, na kterou protějšek B zasílá UDP pakety, které server zabalí do TURN zpráv a přepošle klientu A. Mechanismus a protokol TURN je popsán v RFC 5766 – Traversal Using Relays around NAT.

Pokud BabelApp klient detekuje pomocí STUN oboustranně symetrický NAT, tak použije TURN server jako reléovou stanici pro přeposílání UDP paketů. Přeposílání paketů je ve srovnání s přímou komunikací méně efektivní vzhledem k šířce pásma (pro TURN server) a celkové latenci (pro koncové zařízení). TURN je proto používaný jako poslední východisko, pokud nelze použít jiný mechanismus pro přímou komunikaci.

BabelApp implementuje STUN server společně s TURN serverem a obě tyto služby musí být spuštěny na počítači s veřejnou internetovou adresou.

2.1.6 CENTRÁLNÍ ADRESÁŘ KONTAKTŮ A UŽIVATELSKÉ ADRESÁŘE: ADDRESS BOOK

Kontakty jsou odesilatelé a příjemci zpráv, jejich veřejné klíče jsou základem kryptografického modelu pro šifrování zpráv.
Skupiny zajišťují organizaci kontaktů podle různých potřeb, například podle organizační struktury, projektů nebo zákazníků. Každý kontakt může být členem libovolného množství skupin. Pro nastavení oprávnění ke skupinám jsou vytvářeny role, které mohou mít nastavena práva k jedné nebo více skupinám. Uživatelé získávají práva všech rolí, do kterých jsou přiřazeni. V současné době jsou implementována 2 práva:

- Právo **„List“** pro nalezení kontaktu ve skupině zadáním masky jména
- Právo **„Add“** pro přidání skupiny do uživatelského adresáře.

Uživatelské adresáře jsou vytvářeny na serveru a synchronizovány na všechna zařízení uživatele. Každý uživatel může do svého adresáře přidat libovolné kontakty, které vyhledal s právem „List“, nebo kontakty, které zadal kompletním BabelApp jménem (jméno#babelapp_server). Uživatelské adresáře eliminují extenzivní množství kontaktů, které by bylo nutné synchronizovat na lokální zařízení a současně umožňují efektivně vyhledávat a používat kontakty, s kterými se často komunikuje.

Edice BabelApp pro veřejné nekomerční použití vyžaduje pro přidání kontaktu do uživatelského adresáře zadání celého BabelApp jména kontaktu (nerozlišuje přitom malá a velká písmena a diakritiku) a nepodporuje vyhledávání kontaktů pomocí masky jména.

2.1.7 KOMUNIKAČNÍ BRÁNA: API GATEWAY

Obousměrné zasílání zpráv a dokumentů šifrovaných mezi koncovými body přenosu vyžaduje kompletní implementaci BabelApp protokolu, která je k dispozici v rámci klientů pro mobilní zařízení, PC a Mac. Aby bylo možné využít mechanismus rozesílání (jednosměrná komunikace) zašifrovaných zpráv a dokumentů i z širokého spektra aplikací a inteligentních programovatelných zařízení, obsahuje edice BabelApp komunikační bránu s jednoduchým REST API rozhraním. Brána umožňuje snadnou integrace serveru s aplikacemi třetích stran pro rozesílání šifrovaných zpráv a dokumentů na zařízení s BabelApp klientem.

Integrovaná aplikace má speciální účet na serveru, tzv. API kontakt. Privátní klíč AK aplikace je uložen v zašifrované formě v úložišti komunikační brány, která implementuje klientskou část BabelApp protokolu a šifruje za aplikaci odesílaná data. Aplikace obdrží od správce jméno API kontaktu a náhodně vygenerované autentizační heslo složené z abecedy: {12346789ABCDEFGHJKMNPQRTUVWXY abcdefghjkmnpqrtuvwx}, o délce 128 bitů.

Pro bezpečné uložení privátního D-H klíče aplikace AK se vygeneruje náhodná sůl a z hesla aplikace a hodnoty soli je s použitím PBKDF2 vygenerován šifrovací klíč BK s délkou 128 bitů:

BK = PBKDF2[hmacWithSHA1]
(heslo, sůl, počet_iterací, 128)

Počet iterací: 1 000

Privátní klíč AK je kódován ve formě ASN.1 podle specifikace PKCS#8 (RFC 5208) a výsledná struktura je opatřena zarovnáním PKCS#7 padding.

Pomocí klíče BK s použitím algoritmu AES v CBC módu je zašifrován privátní klíč aplikace AK. Inicializační vektor má hodnotu 0.

eAK = ENC-CBC[BK,0](AK)

Takto šifrovaný privátní klíč eAK je společně s hodnotou soli uložen do databáze ve formě ASN.1 struktury.

Aplikace komunikuje s bránou prostřednictvím SSL kanálu, pro autentizaci aplikace je použit mechanismus úspěšného rozšifrování privátního klíče AK. Jakmile má brána k dispozici rozšifrovaný privátní klíč aplikace AK, může šifrovat zprávy aplikace stejným způsobem, jako klient BabelApp, viz 3.5.6.

2.1.8 PLÁNOVAČ PRO ROZESÍLÁNÍ ZPRÁV SERVEREM: MESSAGE SCHEDULER

Server umožňuje naplánovat a automaticky rozesílat zprávy uživatelům. Zprávy lze rozesílat okamžitě nebo v plánovaném čase pro vybranou množinu příjemců, nebo v určenou dobu, která uplyne po registraci zařízení. Ke zprávám mohou být přiloženy přílohy.

Rozesílání zpráv ze serveru využívá technologii komunikační brány, viz 2.1.4.

2.1.9 BRÁNA PRO ZASÍLÁNÍ UPOZORNĚNÍ: PUSH NOTIFICATION GATEWAY

Mobilní zařízení se systémy iOS a Android umožňují příjem aplikačních push notifikací, které upozorňují na zprávu čekající na serveru na doručení, i když klientská aplikace BabelApp běží na pozadí, nebo není spuštěna.

Push notifikace jsou zařízením rozesílány v rámci infrastruktury Apple/Google notifikačních serverů.

Pro zasílání požadavků k rozeslání push notifikací prostřednictvím Apple/Google notifikačních serverů slouží BabelApp Push Notification Gateway. Zaslání požadavků je volitelné a správce je může povolit, nebo zakázat. Požadavek na zaslání push notifikace musí být elektronicky podepsán s použitím privátního klíče, který má k dispozici pouze Push Notification Gateway a certifikátu, který je registrován u Apple/Google notifikačních serverů.

Každý BabelApp server je vybaven privátním klíčem a certifikátem, vydaným certifikační autoritou společnosti Oksystem a.s., pro podpis požadavků, které jsou zasílány na Push Notification Gateway, kde se po ověření podpisu vytvoří a podepíše nový požadavek na push notifikaci za příslušné mobilní zařízení a zašle na příslušný Apple/Google notifikační server.

Požadavky na zaslání push notifikací a vlastní notifikace neobsahují žádná data o přenášených zprávách.

2.1.10 WEB STRÁNKY ADMINISTRÁTORA ADMIN CONSOLE

Administrace BabelApp serveru se provádí prostřednictvím web stránek, které jsou publikovány na adrese https://babel.doména:port. Doména je tvořena DNS jménem domény organizace, v které je umístěn BabelApp server. Implicitní port je 9091, jiná adresa portu může být zvolena při instalaci. Server musí mít nainstalován certifikát pro tuto doménu, certifikát je vydán certifikační autoritou společnosti OKsystem a.s. na základě žádosti vytvořené správcem.

2.1.11 WEB STRÁNKY UŽIVATELE: CLIENT DASHBOARD

Uživatelé, kteří mají vytvořen účet na BabelApp serveru, mají k dispozici osobní stránku, na které mohou zjistit stav svých zařízení a požádat o připojení nového zařízení. Požadavek musí být schválen správcem, následně se na klientské stránce zobrazí QR kód s OTP pro připojení nového zařízení.

Správce může nastavit URL pro přístup k osobní stránce a aktivovat autentizaci prostřednictvím SSO.

2.2 KOMUNIKACE MEZI SERVERY

Každý klient může mít účet na jednom, případně na více BabelApp serverech, což mu umožňuje přímo komunikovat s uživateli registrovanými na těchto serverech. Aby bylo možné zasílat zprávy i příjemcům, kteří mají účty na jiných serverech, než odesilatel, implementuje BabelApp komunikaci mezi servery.

Pokud má uživatel účty na více serverech, jeden z nich zprostředkovává komunikaci mezi servery.

2.2.1 BABELAPP JMÉNO

Každý uživatel je identifikován v síti BabelApp pomocí jednoznačné adresy (BabelApp jméno):

jméno#babelapp_server

kde: **Jméno** – jméno uživatele (kontaktu) v databázi účtů BabelApp serveru.

babelapp_server – plně kvalifikované DNS jméno (FQDN) BabelApp serveru, pro který je společností OKsystem vystaven certifikát.

Pro jméno babelapp_server musí existovat DNS záznam typu A.

Pro komunikační porty Babelapp serveru se použijí SRV záznamy, pokud neexistují, pak se použijí implicitní porty. Implicitní port pro připojení klientů k BabelApp serveru je 5222.

Příklad:

BabelApp server s FQDN babel.oksystem.cz je dostupný na internetové adrese 193.222.130.33

TCP port pro připojení klientů k tomuto serveru ke službě _babel je 5222

Odpovídající DNS záznamy jsou:

babel.oksystem.cz A 193.222.130.33

_babel._tcp.oksystem.cz. 86400 IN SRV 10 10 5222 babel.oksystem.cz

2.2.2 REGISTRACE MEZI SERVERY

Pro komunikaci mezi servery je nezbytná registrace účtů serverů, podobně, jako jsou registrovány účty uživatelů. Správce serveru má možnost zakázat registraci vyjmenovaných, nebo všech serverů a neumožnit komunikaci s těmito servery.

Pokud je potřeba, aby se server S1 zaregistroval k serveru S2, odešle S1 serveru S2 registrační zprávu s typem SERVER. Server S2 kontroluje, zda S1 vlastní certifikát podepsaný společností OKsystem a vydaný pro doménu udanou v registraci. Pokud ano, S2 zkontroluje, zda správce nezakázal server S1 zaregistrovat.

2.2.3 SYNCHRONIZACE KONTAKTŮ MEZI SERVERY

Registrovaný server S1 si může vyžádat od S2 aktualizaci svého adresáře (Address Book) o data kontaktu zaregistrovaného na S2, aby získal jeho veřejný klíč a mohl zašifrovat metadata o zprávách k doručení tomuto kontaktu.

Server S1 může zasílat synchronizovaným kontaktům registrovaným na S2 zprávy s metadaty o zprávách, které lze stáhnout přímo ze serveru S1.

2.2.4 ODESLÁNÍ A DORUČENÍ ZPRÁVY

Zařízení uživatele U1#S1 odešle zprávu Z na svůj implicitní BABEL server S1. Server S1 zprávu Z uloží a vytvoří novou zprávu M, která obsahuje údaje o odesílateli, konverzaci a platnosti zprávy (metadata zprávy). Cílový server zprávu M doručí, klient příjemce U2#S2 z ní získá URL a identifikátor zprávy Z. Klient U2#S2 se připojí na dané URL a autentizuje se tak, že prokáže, že vlastní privátní klíč, kterým je možné zprávu Z dešifrovat a zprávu Z stáhne.

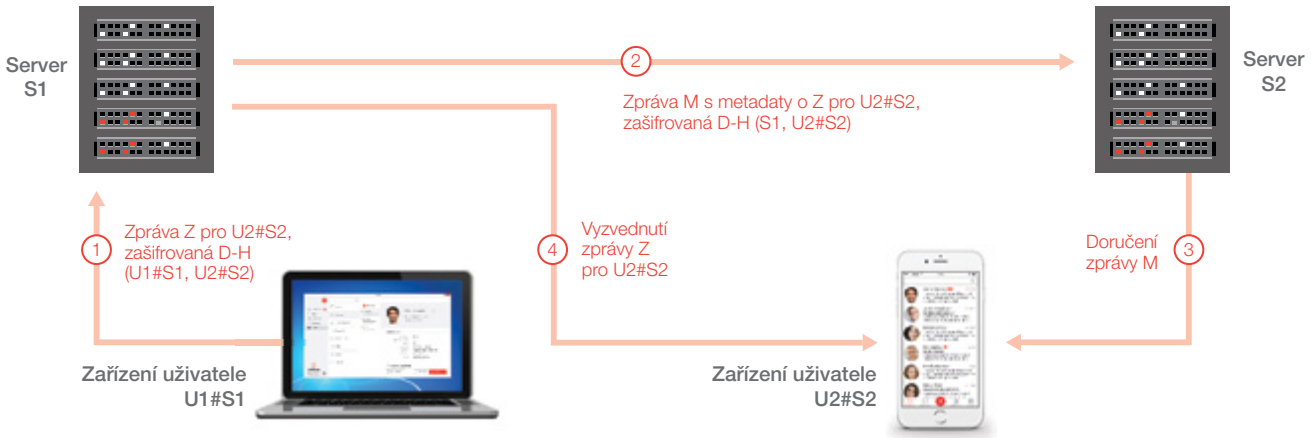


Schéma pro odeslání a doručení zprávy mezi servery.

2.3 UMÍSTĚNÍ V SÍTI A KOMUNIKACE

2.3.1 BABELAPP SERVER

BabelApp server je typicky umístěn v rámci tzv. „demilitarizované zóny" firemní sítě, která je propojena směrovači a firewallovou soustavou s interní sítí a internetem. Směrovače mohou používat překlad adres (NAT) pro přístup do internetu. Nastavení firewallů musí umožnit prostupy pro kombinaci protokol/ip adresa/port mezi BabelApp serverem v DMZ a interní sítí respektive internetem. Adresy a porty uvedené v následujícím obrázku jsou pouze pro účely demonstrace, v konkrétní instalaci mohou být odlišné.

2.3.2 VIRTUÁLNÍ BABELAPP SERVERY A SNI

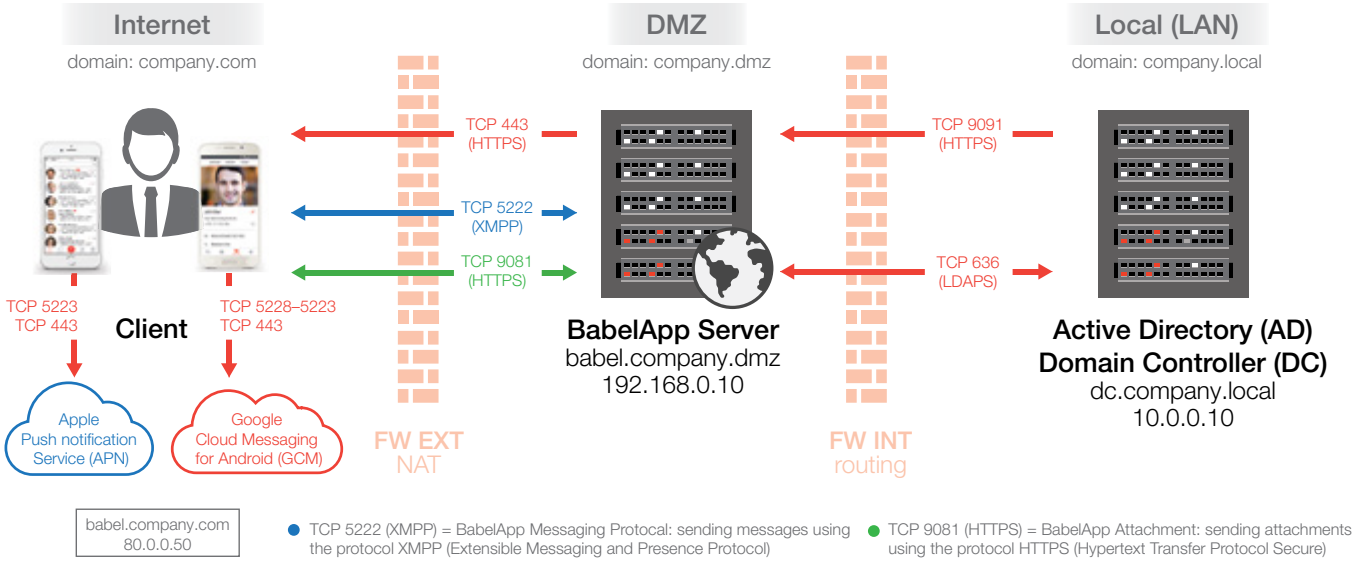
Server i klienti BabelApp v rámci TLS komunikace podporují technologii SNI (Server Name Indication), tedy provoz více virtuálních BabelApp serverů/domén, sdílejících jednu veřejnou IP adresu. SNI je založen na mechanismu rozšíření TLS, v rámci kterého BabelApp klient při navazování TLS relace předává TLS serveru DNS jméno virtuálního BabelApp serveru, na který je komunikace přesměrována.

Podpora technologie SNI umožňuje hostování BabelApp serverů v cloudu.

Technologie SNI je popsána v RFC 4366 a RFC 6066.

2.3.3 STUN/TURN SERVER

STUN/TURN server slouží pro službu VoIP (viz 2.1.4 a 2.1.5) a musí být vybaven veřejnou internetovou adresou. TURN server při provozu otevírá větší množství portů a je proto vhodné, aby byl umístěn v samostatné bezpečnostní zóně DMZ, která umožní potřebné prostupy.



2.3.4 KLIENTI BABELAPP

BabelApp klient je k dispozici pro všechny významné platformy mobilních i stolních zařízení, viz 1.2.1. Uživatel může mít pod svým účtem registrováno více zařízení.

Každý BabelApp klient implementuje kompletní aplikační protokol pro šifrovanou komunikaci mezi BabelApp klienty a šifrované uložení odeslaných a přijatých zpráv a dokumentů na zařízení.

BabelApp klienti pro mobilní zařízení se systémy Android a iOS implementují klientskou část signalizačního protokolu pro VoIP a šifrovanou hlasovou komunikaci založenou na SRTP protokolu (SRTP je profil RTP). V rámci RTP komunikují mobilní klienti přímo, pokud to konfigurace sítě umožňuje, nebo prostřednictvím reléování s využitím TURN serveru.

BabelApp klient komunikuje s BabelApp servery prostřednictvím mobilního datového připojení k internetu, nebo firemní LAN/wifi. Pro komunikaci se serverem musí být zařízení registrováno k serveru pod účtem uživatele a mít nastaveno DNS jméno BabelApp serveru a komunikační port.

BabelApp klient může mít konfigurováno připojení k více BabelApp serverům, jeden z nich je implicitní. Implicitní server pro uživatele zprostředkovává komunikaci mezi servery.

3. KRYPTOGRAFICKÝ NÁVRH

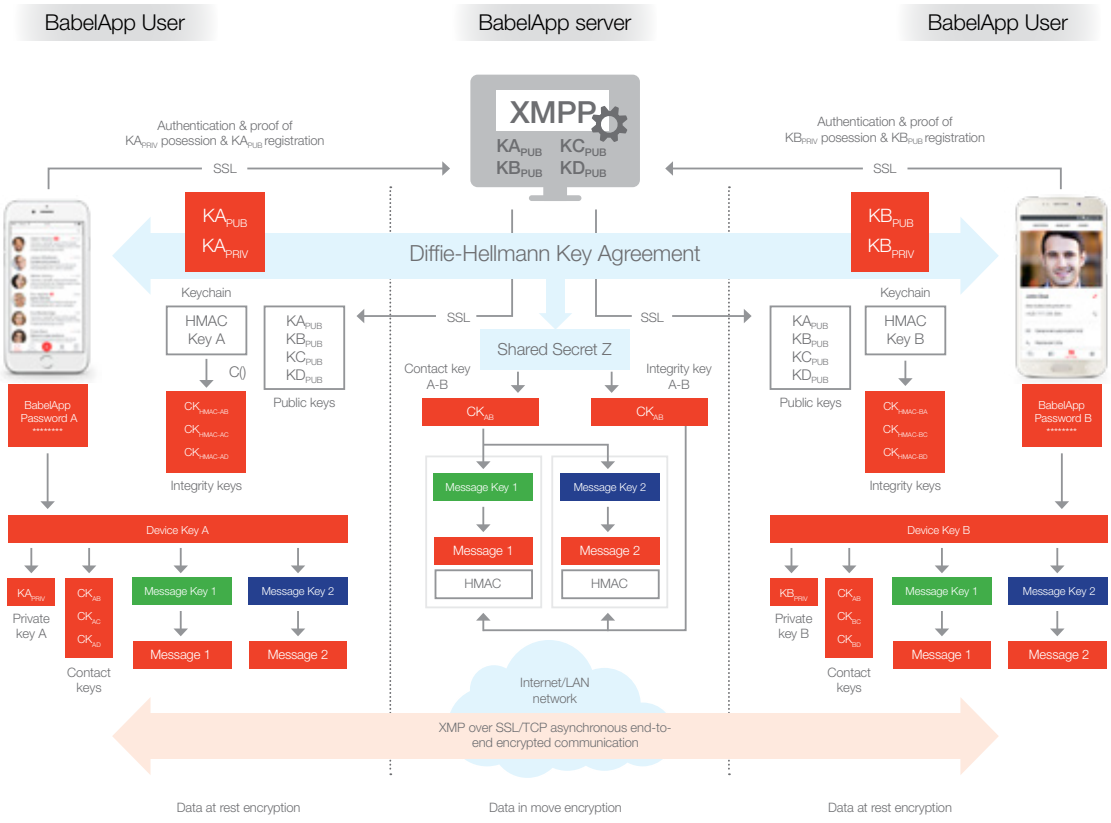
3.1 VÝCHODISKA PRO KRYPTOGRAFICKÝ NÁVRH

Při kryptografickém návrhu se vycházelo z následujících požadavků:

- Cílem je ochrana privátnosti a obchodního tajemství, které je obsahem komunikace, nikoli utajení skutečnosti, že komunikace proběhla.
- Používá se aplikační šifrování mezi koncovými body přenosu pro přenos zpráv a dokumentů.
- Používá se aplikační šifrování při uložení zpráv a dokumentů na zařízeních.
- Používá se aplikační šifrování mezi koncovými body přenosu při VoIP telefonování.
- Nepoužívají se žádné certifikáty veřejného klíče uživatelů ani zařízení.
- Používá se server pro správu uživatelů a zařízení, distribuci a synchronizaci veřejných klíčů a zprostředkování asynchronní komunikace mezi zařízeními uživatelů pro přenos zpráv, dokumentů, upozornění a VoIP signalizace. Server nemá žádné klíče pro dešifrování přenášených zpráv. Server má pouze přístup k informacím o uživatelích, zařízeních a metadatech přenášených zpráv. Přenášené šifrované zprávy nejsou ukládány na serveru po uplynutí časového limitu pro doručení. Server je zpravidla pod vlastní správou organizace.
- Každý uživatel může mít více zařízení (smartphone, tablet, PC, notebook…), zpráva je odeslána vždy z jednoho zařízení, nicméně je třeba zajistit synchronizaci odeslaných a přijatých zpráv na všech zařízeních uživatele.
- Používají se výhradně standardní silné kryptografické algoritmy, doporučené parametry a módy operací.
- Používají se techniky pro eliminaci aktivních útoků – kontrola integrity, autenticity a pořadí zpráv před jakýmkoli pokusem o dešifrování přijatých zpráv.

3.2 KRYPTOGRAFICKÝ MODEL

Na následujícím schématu je uvedeno konceptuální schéma kryptografického modelu. Model popisuje hierarchický systém klíčů a aplikační šifrování dat při zasílání zpráv a uložení zpráv na zařízeních. Rozšíření krytografického modelu pro VoIP je uvedeno v článku 3.7.2.
Kromě aplikačního šifrování je navíc použit protokol TLS pro veškerou komunikaci mezi klienty a serverem a systémové služby pro šifrování dat a databáze na zařízení uživatele.



3.3 KRYPTOGRAFICKÉ ALGORITMY

BabelApp používá následující kryptografické algoritmy:

- Diffie-Hellman podle NIST SP 800-56A
- AES 128, AES 256 podle odstavce 5 FIPS PUB 197 v CBC, ECB a CTR módech podle NIST SP 800-38A
- PBKDF2 podle PKCS#5 v2
- SHA-2 podle FIPS PUB 180-4, odstavec 6.2 s délkou otisku 256 bitů
- HMAC podle v RFC 2104 a rozšíření pro použití hashovacích algoritmů SHA2 podle RFC 4868

Uvedené algoritmy jsou použity v dále popsaných kryptografických schématech a protokolech.

3.4 KRYPTOGRAFICKÉ PROTOKOLY

S využitím kryptografických algoritmů a klíčů jsou realizovány dále uvedené standardní protokoly a aplikační šifrování při komunikaci a při uložení dat na zařízeních.

3.4.1 KOMUNIKACE MEZI KLIENTY A SERVERY BABELAPP

3.4.2 PROTOKOL TLS

Komunikace mezi klienty a servery Babel probíhá v rámci protokolu Transport Layer Security TLS v1.2 podle RFC5246. K navázání TLS komunikace je server vybaven certifikátem vydávaným společností OKsystem, na všech platformách je tento certifikát součástí kódu klienta Babel pro striktní kontrolu autenticity serveru.

V rámci TLS spojení jsou zasílány JSON zprávy s kontrolou integrity (viz 3.5.7), přenášející (kromě jiného) zprávy a přiložené dokumenty aplikačně zašifrované mezi koncovými body přenosu (viz 3.5.6 resp. 3.5.9), nebo signalizační zprávy pro VoIP.

3.4.3 PROTOKOL SRTP

Telefonická komunikace VoIP používá transportní protokol UDP, přenášející aplikačně šifrovaný datový proud založený na protokolu SRTP (RFC3711), který je profilem protokolu RTP (RFC3550).

3.4.4 REGISTRACE KLIENTA K BABELAPP SERVERU POMOCÍ OTP

Registrace zařízení klienta používá autentizaci s využitím One Time Password Protocol, který je založený na náhodně generovaném hesle a odvození otisku hesla pomocí PBKDF2 podle NIST sp800-132.

3.4.5 REGISTRACE KLIENTA K BABELAPP SERVERU POMOCÍ AD SSO

BabelApp klient umožňuje využít alternativní registraci firemních uživatelů, kteří mají účet v LDAP adresáři (nejčastěji v Active Directory), na základě systému jediného přihlášení.

3.4.6 PODÁNÍ DŮKAZU O DRŽENÍ D-H PRIVÁTNÍHO KLÍČE

Důkaz o držení privátního klíče k předložené hodnotě veřejného klíče je proveden při registraci zařízení klienta k BabelApp serveru na základě protokolu Diffie-Hellman Proof of Possession podle RFC 6955.

3.4.7 AUTENTIZACE KLIENTA K BABELAPP SERVERU

Na základě registrace zařízení klienta k serveru BabelApp se na serveru vytvoří náhodné autentizační heslo AH a jeho uloží 160 bit autentizační otisk AS odvozený pomocí PBKDF2:

AS = PBKDF2[hmacWithSHA1]
(heslo_AH, sůl, počet_iterací, 160)

Autentizační heslo je zasláno v rámci TLS relace na zařízení klienta pro následné autentizace klienta k serveru.

Klient BabelApp se automaticky autentizuje k serveru v okamžiku spuštění, za předpokladu, že je k dispozici internetové spojení se serverem. Pro autentizaci klienta k serveru není požadována předchozí autentizace uživatele ke klientu.

3.4.8 GENEROVÁNÍ KLÍČŮ

Generování náhodného kryptografického materiálu (šifrovacích klíčů, autentizačních klíčů, DH klíčových párů) je založeno na generátorech náhodných čísel závislých na platformě klienta.

3.4.9 ODVOZENÍ KLÍČŮ Z HESLA

Pro odvození klíče z hesla je použita funkce PBKDF2 podle PKCS#5 v2, využívající pseudonáhodnou funkci HMAC with SHA256 a vysoký počet iterací.

3.5 APLIKAČNÍ ŠIFROVÁNÍ PŘI KOMUNIKACI

Aplikační šifrování je základem bezpečnosti při komunikaci mezi BabelApp klienty a při uložení dat na klientech. Aplikační šifrování je založeno na kombinaci algoritmů symetrické kryptografie s tajnými klíči a nesymetrické kryptografie s veřejnými klíči. Dále je podrobněji popsána implementace kryptografického modelu 3.2.

3.5.1 NALEZENÍ SHODY NA KLÍČI CONTACT KEY

K nalezení shody na klíči je použit protokol Diffie-Hellman, popsaný v RFC 2631 a ve standardech ISO 14883-3, ANSI X9.62 a NIST SP 800-56A.
Implementace protokolu Diffie-Hellman v BabelApp používá celočíselnou multiplikativní grupu modulo p, s parametry p, g, a q podle RFC 5114, odstavec 2.3:
bitová délka prvočíselné multiplikativní grupy řádu p je 2048-bit
bitová délka prvočíselné multiplikativní podgrupy generované g řádu q: 256-bit

3.5.1.1 DOMÉNOVÉ PARAMETRY

Hodnoty doménových parametrů p, q, q (v šestnáctkové soustavě):

p = 87A8E61DB4B6663CFFBBD19C651959998CEEF608660
DD0F25D2CEED4435E3B00E00DF8F1D61957D4FAF7DF45
61B2AA3016C3D91134096FAA3BF4296D830E9A7C209E0
C6497517ABD5A8A9D306BCF67ED91F9E6725B4758C022
E0B1EF4275BF7B6C5BFC11D45F9088B941F54EB1E59BB
8BC39A0BF12307F5C4FDB70C581B23F76B63ACAE1CAA
6B7902D52526735488A0EF13C6D9A51BFA4AB3AD834
7796524D8EF6A167B5A41825D967E144E5140564251CCAC-
B83E6B486F6B3CA3F7971506026C0B857F689962856DE
D4010ABD0BE621C3A3960A54E710C375F26375D7014103
A4B54330C198AF126116D2276E11715F693877FAD7EF09
CADB094AE91E1A1597

g=3FB32C9B73134D0B2E77506660EDBD484CA7B18F21
EF205407F4793A1A0BA12510DBC15077BE463FFF4FED-
4AAC0BB555BE3A6C1B0C6B47B1BC3773BF7E8C6F629
01228F8C28CBB18A55AE31341000A650196F931C77A57F2
DDF463E5E9EC144B777DE62AAAB8A8628AC376D282D6ED
3864E67982428EBC831D14348F6F2F9193B5045AF2767164
E1DFC967C1FB3F2E55A4BD1BFFE83B9C80D052B985D182
EA0ADB2A3B7313D3FE14C8484B1E052588B9B7D2BBD2DF-
016199ECD06E1557CD0915B3353BBB64E0EC377FD02837
0DF92B52C7891428CDC67EB6184B523D1DB246C32F6307
8490F00EF8D647D148D47954515E2327CFEF98C582664B-
4C0F6CC41659

q=8CF83642A709A097B447997640129DA299B1A47D1EB-
3750BA308B0FE64F5FBD3

3.5.1.2 GENEROVÁNÍ DH KLÍČŮ

Každý uživatel BabelApp vygeneruje na svém zařízení privátní klíč X (náhodné číslo 1 < X < q - 1), který je tajný. Korespondující veřejný klíč je spočten jako Y = g^X mod p

Poznámka 1:
Uživatel generuje DH klíčový pár na každém zařízení, které registruje ke svému účtu na serveru BabelApp. Pokud již dříve bylo k serveru registrováno toto, nebo jiné zařízení, může uživatel přepsat původní klíče novými, odstranit dříve registrovaná zařízení a revokovat původní klíč, nebo přenést původní DH klíč na nově registrované zařízení. Podrobný popis pro bezpečný přenos klíče mezi zařízeními je uveden v článku 3.5.11.

Poznámka 2:
Privátní klíč X je na zařízení zašifrován pomocí klíče zařízení DK, jak je popsáno v článku 3.6.5.

Příklad 1:
Uživatel A vygeneruje náhodný privátní klíč Xa a spočítá veřejný klíč Ya s využitím doménových parametrů (p, q, g) a Ya = g^Xa mod p

Xa = 1973D625770FCD1058C9474213DD3E7905DEAA4
A2226D2FA26A08504D8700ABF

Ya = 3415B6FD8C2531D0DDC5E38A7FFD6E0D03EB5446
94B6A74D0B768B57D1CEC6A9D8B18F6BB920B6382ED
FCF113DDEB549E5BC272F031984514C2F87435A7B0668
F850A21B82E2C1EDADE3D2BE7358BB97859B0A22B0134
AB457C77EC6D3308188C197BD4D8FFE4D698DC4805715
E049043290B2EE11D9F8F834326F11460C56A743925D29
C1A862D51BF13556F1D9A44A1D1EDC80E78052AC97A4A
2D998045EFEDCF6C3463D88C69BA9CA904CD53BA5E0D7
6313A29FC1CB307385FE047B7FBD176C1BB35D57F9B7C4
AA63B670164BA59A4EB7F1ACB525C55F74C454E0F3888F
79152A7127E6976CCEF821391591005551064217193538
448317302609964318259122758009CEF82139159100555
106421719353844831730260996431825912275800 9

3.5.1.3 REGISTRACE HODNOTY VEŘEJNÉHO KLÍČE NA SERVERU

Součástí registrace prvního zařízení k účtu uživatele na serveru je registrace hodnoty veřejné části DH klíče uživatele, který byl vygenerován při instalaci BabelApp na zařízení. Registrace je podmíněna úspěšnou autentizací uživatele, která v závislosti na nastavení serveru může být provedena pomocí zadání jména a jednorázového hesla (většinou s využitím QR kódu), nebo založena na autentizaci k firemnímu LDAP adresáři.

Po přihlášení musí zařízení uživatele podat serveru kryptografický důkaz držení privátní části DH klíče k předložené hodnotě veřejného klíče. Tento důkaz je založen na protokolu Diffie-Hellman Proof of Possession podle RFC 6955.

Po úspěšné registraci vrátí server klientovi persistentní autentizační údaje pro následná přihlášení.

Registrované veřejné klíče uživatelů poskytuje server všem zařízením, která splňují podmínky pro viditelnost uživatele v seznamu kontaktů na serveru a vyžádali si synchronizaci kontaktu na zařízení.

3.5.1.4 NALEZENÍ HODNOTY SDÍLENÉHO TAJEMSTVÍ Z MEZI UŽIVATELI A a B

Předpoklady:
Uživatelé A a B sdílí doménové parametry (p, q, g) a získali důvěryhodným způsobem veřejné klíče protistrany

Uživatel A má privátní klíč Xa a veřejný klíč Ya
Uživatel B má privátní klíč Xb a veřejný klíč Yb

Výpočet:
Uživatel A vypočte hodnotu sdíleného tajemství
Z = Yb^Xa mod p
Uživatel B vypočte hodnotu sdíleného tajemství
Z = Ya^Xb mod p

Příklad 2:
Uživatel B spočítá hodnotu sdíleného tajemství Z na základě hodnoty svého privátního klíče Xb a veřejného klíče uživatele A Ya pomocí Z = Ya^Xb mod p

Xb = 7433D90B61EA231F2350ADE584EE047D-
DC8116D077BB6B6977CAAE2DDE399545

Z = 1D0D34D49D5F613892611DC620D66D80F2690249D
474248B4CA4863D2E5EA2F722E9C98B91D70BC0E791BC3
AFB5F105F518E749FDAC9A0374DD340B8D369409BAB061
EEE708672F6954883F4D21311A5331E6DABA2E4EB620DB
DCA8343F7033E8BB3C1929DD406250D4E4AAECE1B063
CABB5B82966B53AFBA82DEEFC969D24888B44CCE8ECCC
3F5F1BE3D4CE1AC5E10A9F38121ABDA08F55301E5495A6
AD78C80562E501DFA51DEACEEDFB5698722AC9FE7B971
6F60956C0B06EA069C91AFBC26C07831F7FF0764F2818
A54FFD849D4E679BD4F580E20CBD3FAB0BE831020C67CC
4350B996DDDCFC8D88021AF0804C4099C3A04676E52A
EAA13A7212A3640AF2

3.5.2 ODVOZENÍ KLÍČOVÉHO MATERIÁLU Z HODNOTY SDÍLENÉHO TAJEMSTVÍ Z

Pro odvození klíčového materiálu (klíče kontaktu a klíče pro kontrolu integrity) je nutné použít kryptografickou funkci pro generování klíčů potřebné délky z hodnoty Z. Je použit algoritmus uvedený v RFC 2631, odstavec 2.1.2

Z hodnoty Z lze generovat prakticky libovolné množství klíčového materiálu KM pomocí algoritmu:

KM = H(Z || OTHER_INFO)
Kde H je hashovací funkce a OTHER_INFO je ASN.1/DER struktura

3.5.2.1 Klíč kontaktu

Klíč kontaktu CK má délku 128 bitů (AES).
Použitá hashovací funkce H je SHA1, z hodnoty hashe o délce 160 bitů se použije počátečních 128 bitů pro hodnot klíče CK

Struktura OTHER_INFO je následující:

```
static uint8_t DER_OTHER_INFO[] = {
0x30, 0x1B, // sequence length 27
 // keyInfo
0x30, 0x11, // sequence length 17
0x06, 0x09, 0x60, 0x86, 0x48, 0x01, 0x65, 0x03,
0x04, 0x01, 0x02,
// algorithm OID 2.16.840.1.101.3.4.1.2 AES 128 CBC
0x04, 0x04, 0x00, 0x00, 0x00, 0x00,// counter
0xA2, 0x06, 0x04, 0x04, 0x00, 0x00, 0x00, 0x80 //
suppPubInfo EXPLICIT
};
```

Příklad 3:
CK = sha1(Z || '30 1B 30 11 06 09 60 86 48 01 65 03 04 01 02 04 04 00 00 00 00 A2 06 04 04 00 00 00 80')
Hodnota klíče CK pro sdílené tajemství Z uvedené v Příkladu 2:
CK = 5FAD22F98B27648BE55F0B40D58E4FA0

3.5.2.2 Klíč pro kontrolu integrity

Klíč pro kontrolu integrity CK_{hmac} má délku 256 bitů.

Algoritmus pro kontrolu integrity zprávy je HMAC with SHA256,
OID = {iso(1) member-body(2) us(840) rsadsi(113549) digestAlgorithm(2) hmacWithSHA256(9)}
Použitá hashovací funkce H je SHA256, celá hodnota hashe se použije pro hodnotu klíče CK_{hmac}.

Struktura OTHER_INFO je následující:

```
static uint8_t DER_OTHER_INFO_HMAC[] = {
0x30, 0x1A, // sequence length 26
// keyInfo
0x30, 0x10, // sequence length 16
0x06, 0x08, 0x2A, 0x86, 0x48,  0x86, 0xF7, 0x0D,
0x02, 0x09,
// algorithm OID 1.2.840.113549.2.9 – PBKDF2 HMAC
with SHA256
0x04, 0x04, 0x00, 0x00, 0x00, 0x01, // counter
0xA2, 0x06, 0x04, 0x04, 0x00, 0x00, 0x01, 0x00 //
suppPubInfo EXPLICIT
};
```

Příklad 4:
Hodnota klíče CK_{hmac} pro sdílené tajemství Z uvedené v Příkladu 2:

CK_{hmac} = F180BE4E4196A7FE9AA3090046F732DFC
12D2526F0CC4E3996D842F5230909F2

3.5.3 GENEROVÁNÍ KLÍČŮ ZPRÁVY

Pro každou zprávu se generuje náhodný klíč MK = RND(16) o délce 128 bitů. Pro generování klíčů se používají generátory náhodných čísel závislé na platformě.

3.5.4 ZAROVNÁNÍ DAT (PADDING)

Před šifrováním jsou data zarovnána na celistvý násobek 16 oktetů (byte padding) způsobem podle PKCS#7, které je popsáno v RFC 5652, článek 6.3. – zpráva se zarovná zřetězením s (16 – (délka_zprávy mod 16)) oktety s hodnotou (16 – (délka_zprávy mod 16)).

Příklad 5:
Pokud mají data délku 1 byte, zřetězí se s 15 bajty '0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F '

Pokud mají data délku 16 byte, zřetězí se s 16 bajty '10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10'

Pokud mají data délku 26 byte, zřetězí se s 6 bajty '06 06 06 06 06 06'

3.5.5 ŠIFROVÁNÍ KLÍČŮ

Pro šifrování klíčů zprávy MK o délce 128 bitů se používá algoritmus AES-128 (FIPS PUB 197) v módu ECB podle NIST SP 800-38A, článek 6.1.

Před šifrováním klíče není použit padding, klíč má fixní délku 128 bitů.

V módu ECB se nepoužívá inicializační vektor.

eMK = ENC-ECB[CK](MK)

3.5.6 ŠIFROVÁNÍ ZPRÁV

3.5.6.1 PŘÍPRAVA ZPRÁVY

Všechny textové zprávy T se převádějí do znakové sady Unicode s kódováním UTF-8 podle RFC 3629, aby bylo dosaženo kompatibility textu mezi platformami.

Před šifrováním zprávy se získá časová značka TS (počet milisekund od půlnoci 1.1. 1970 UTC). Tato časová značka se kóduje na TS', 4 oktety s organizací bitů „big endian", aby bylo dosaženo kompatibility mezi platformami. TS' se připojí na konec zprávy.

Zpráva s připojenou časovou značkou TS' se zarovná způsobem podle PKCS#7.

3.5.6.2 ŠIFROVÁNÍ ZPRÁVY

Pro každou zprávu T se generuje náhodný klíč MK = RND(16) o délce 128 bitů.

Pro šifrování textu zprávy se používá algoritmus AES-128 (FIPS PUB 197) v módu CBC podle NIST SP 800-38A, článek 6.2.

Inicializační vektor IV má hodnotu 0 (IV = '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00') pro všechny zprávy. Použití shodné hodnoty IV pro všechny šifrované zprávy není problém vzhledem k tomu, že každá zpráva je šifrována jiným, náhodně generovaným klíčem.

eT = ENC-CBC[MK,IV](utf8(T) || TS')

Příklad 6:

- T je otevřený text zprávy
- MK je klíč pro šifrování zprávy
- CK je klíč kontaktu spočtený v Příkladu 3
- eMK je klíč pro šifrování zprávy zašifrovaný klíčem kontaktu CK
- TS' je kódovaná časová značka
- eT je zašifrovaná zpráva

T = "Privacy exists"
utf8(T) = 507269766163792065786973747312345678 0E0E0E0E0E0E0E0E0E0E0E0E
MK = 11121314212223243132333441424344
(hex string)
CK = 5FAD22F98B27648BE55F0B40D58E4FA0
eMK = 874B7E0985282C274F63161987BBC09D
TS' = 12345678
utf8(T) || TS' || padding =
507269766163792065786973747312345678 0E0E0E0E0E0E0E0E0E0E0E0E0E0E
eT = B8EB1C3D985D2D92A8C574C85ABE49B7
C1FF8BCAC2B60E1128AF0AB5E2BD0182

3.5.7 ZAJIŠTĚNÍ INTEGRITY ZPRÁV

K zajištění kontroly integrity, autenticity a pořadí doručení šifrovaných zpráv a konverzací, která je nezbytná k detekci a eliminaci aktivních útoků, jsou použity dva nezávislé mechanismy:

- Kryptografický autentizační kód HMAC celé JSON zprávy
- Číslování pořadí odeslaných zpráv

3.5.7.1 AUTENTIZAČNÍ KÓD HMAC

Je použit mechanismus Encrypt-then-MAC. Zašifrovaná data jsou opatřena autentizačním kódem vypočteným pomocí algoritmu HMAC-SHA256, na základě definice HMAC v RFC 2104 a rozšíření pro použití hashovacích algoritmů SHA2 podle RFC 4868.

MAC = H(K XOR opad, H(K XOR ipad, text))

kde

H je hashovací funkce SHA-256, podle FIPS PUB 180-4, odstavec 6.2
Ipad je posloupnost 64 oktetů se stejnou hodnotou 0x36
Opad je posloupnost 64 oktetů se stejnou hodnotou 0x5C
K je CK_{hmac} klíč délky 256 bitů

Poznámka: Původní specifikace HMAC uvedená v RFC 2104 používá hashovací algoritmus MD5 nebo SHA1.

3.5.7.2 IDENTIFIKACE A ČÍSLOVÁNÍ ZPRÁV

Každá zpráva M je v rámci XMPP/JSON zprávy identifikována 256 bitovým identifikátorem messageId, který je náhodně vygenerován.

Pořadí zpráv je číslováno 128 bitovým čítačem sequenceId, který je náhodně inicializován a s každou odeslanou zprávou je inkrementován.

3.5.8 PŘÍJEM A DEŠIFROVÁNÍ ZPRÁV

3.5.8.1 KONTROLA INTEGRITY PŘIJATÉ ZPRÁVY

Před dešifrováním zprávy je provedena kontrola integrity přijaté JSON zprávy M pomocí výpočtu HMAC a porovnáním s příslušnou hodnotou autentizačního kódu zprávy.

$HMAC' = HMAC\text{-}SHA256(CK_{hmac}, M)$

V případě, kdy by nesouhlasila hodnota odeslaného autentizačního kódu a autentizačního kódu vypočteného příjemcem, je detekován aktivní útok na změnu nebo podvržení zprávy a příjemce nebude takovou zprávu dešifrovat (v opačném případě by hrozil únik informace postranními kanály při zobrazení chybových stavů) a zobrazí varování.

Pokud by byla přijata zpráva s neplatným pořadím, je zobrazeno varování.

3.5.8.2 DEŠIFROVÁNÍ ZPRÁVY

Příjemce nejprve rozšifruje klíč MK, kterým byla zpráva zašifrována, pomocí klíče CK, který sdílí příjemce s odesilatelem zprávy.

$MK = DEC\text{-}ECB[CK](eMK)$

Příjemce dešifruje zprávu eT pomocí MK, inicializační vektor IV = 0

$T' = DEC\text{-}CBC[MK,IV](eT)$

Zpráva T se získá oříznutím posledních 4 bajtů T' (časová značka) a následným dekódováním UTF-8.

3.5.9 ŠIFROVÁNÍ PŘÍLOH

Klient BabelApp umožňuje zasílat zašifrované přílohy k textovým zprávám (dokumenty libovolného typu). Přílohy jsou zašifrovány stejným klíčem MK, jako vlastní zpráva. Přílohy jsou odesílány odděleně od zpráv. Součástí zprávy jsou pouze metadata příloh:

- nešifrovaná metadata – identifikátor přílohy, hash přílohy a délka přílohy v bajtech.
- samostatně zašifrovaná metadata – typ, jméno a miniatura (např. náhled fotografie) přílohy.

Zašifrovaná metadata jsou šifrována stejným klíčem MK jako vlastní zpráva.

Každá zpráva může obsahovat metadata více příloh.

Odesilatel vygeneruje náhodný inicializační vektor pro šifrování přílohy:

$IV_{Data} = RND(16)$

Odesilatel zašifruje Data pomocí klíče MK a IV_{Data}, před zašifrovaná data připojí IV_{Data}:

$eData = IV_{Data} \,||\, ENC\text{-}CBC[MK, IVData](Data)$

Odesilatel spočítá hash eData pro kontrolu integrity:

$hash = sha256(eData)$

Odesilatel spočítá délku eData v bajtech:

$length = len(eData)$

Odesilatel nahraje eData na server a obdrží zpět identifikátor ID_{file}.

Odesilatel vygeneruje 3 náhodné inicializační vektory pro šifrovaná metadata:

$IV_{Name} = RND(16)$

$IV_{Type} = RND(16)$

$IV_{Thumbnail} = RND(16)$

Odesilatel zašifruje postupně typ, jméno a miniaturu přílohy pomocí klíče MK a příslušného inicializačního vektoru IV_{Type}, IV_{Name}, $IV_{Thumbnail}$:

$eName = IV_{Name} \,||\, ENC\text{-}CBC[MK, IV_{Name}](utf8encode(Name))$

$eType = IV_{Type} \,||\, ENC\text{-}CBC[MK\ IV_{Type}](utf8encode(Type))$

$eThumbnail = IV_{Thumbnail} \,||\, ENC\text{-}CBC[MK, IV_{Thumbnail}](Thumbnail)$

Odesilatel odešle metadata příloh ID_{file}, eName, eType, eThumbnail, hash a length prostřednictvím JSON zprávy s kontrolou integrity (viz 3.5.7).

3.5.10 DEŠIFROVÁNÍ PŘÍLOH

Příjemce zprávy, která obsahuje metadata příloh, použije identifikátor přílohy ID_{file} a stáhne ze serveru zašifrovaná data přílohy eData.

Příjemce spočítá hash eData a porovná s korespondující hodnotou hash, která je součástí metadat zprávy

$hash' = sha256(eData)$

$hash' == hash$; pokud se hodnoty liší, příjemce přeruší zpracování, přílohu zruší a zobrazí varování

Příjemce oddělí inicializační vektor IV_{Data} z prvních 16 bajtů eData:

$IV_{Data} \,||\, eData' = eData$

a použije klíč zprávy MK k rozšifrování eData' a získání rozšifrované hodnoty Data:

$Data = DEC\text{-}CBC[MK, IV_{Data}](eData')$

3.5.11 PŘENOS KLÍČŮ MEZI ZAŘÍZENÍMI UŽIVATELE

Každý uživatel s účtem na serveru BabelApp může mít několik zařízení registrovaných k tomuto účtu. Všechna zařízení uživatele sdílí hodnotu D-H klíčového páru uživatele.

Pro přehlednost označíme nově registrované zařízení NZ a dříve zaregistrované zařízení SZ.

Uživatel generuje DH klíčový pár na každém zařízení, které registruje ke svému účtu na serveru BabelApp. Pokud již dříve bylo k serveru registrováno toto nebo jiné zařízení, vrátí server při pokusu o registraci chybovou zprávu,

v které zašle GUID kontaktu a veřejnou část DH klíče kontaktu.

Uživatel může zvolit, zdali chce přenést původní DH klíč na nově registrované zařízení (NZ), nebo přepsat původní klíče novými, odstranit dříve registrovaná zařízení (SZ) a revokovat původní klíč.

3.5.11.1 PŘENESENÍ KLÍČE ZE SZ NA NZ

Předpokládá se, že oprávněný uživatel má současný přístup k SZ i NZ a že obě zařízení jsou připojena on-line. Pro přenos klíče je nutné opsat hodnotu autentizačního PIN, zobrazeného na NZ a zadat ji na SZ, útočník by tedy musel mít k dispozici nejen autentizační údaje pro registraci NZ k serveru, ale i obě zařízení.

Pokud uživatel zvolí přenesení klíče na nově registrované zařízení (NZ) z jiného, dříve registrovaného zařízení (SZ), vygeneruje se na NZ autentizační PIN, s použitím sdíleného tajemství Z, spočteného na základě privátní části DH klíčového páru NZ a veřejné části DH klíče SZ, který server zaslal NZ v chybové zprávě. Postup výpočtu hodnoty Z je analogický postupu popsaném v článku 3.5.1.4.

Hodnota PIN o délce 5 číslic je spočtena z hodnoty Z následovně:

PIN = SHA1(Z) mod 100000

Server si vyžádá od SZ přenos DH klíčového páru a v této žádosti pošle veřejnou část DH klíče NZ. SZ zobrazí uživateli informaci o žádosti na přenos klíče a vyžádá si zadání autentizačního PIN, které bylo vygenerováno a zobrazeno na NZ. Uživatel zadá na SZ hodnotu PIN, SZ spočítá hodnotu PIN´ na základě stejných údajů jako NZ a porovná spočtenou hodnotu PIN´ a zadanou hodnotu PIN.

V případě shody (PIN´ = PIN) provede SZ následující kroky:

a) Spočítá sdílené tajemství Z na základě privátní části DH klíčového páru SZ a veřejné části DH klíče NZ. Postup výpočtu hodnoty Z je analogický postupu popsaném v článku 3.5.1.4.
b) Vygeneruje 256 bitový AES šifrovací klíč pro šifrování privátního klíče.
c) Zakóduje DH privátní klíč SZ a jeho identifikátor GUID do ASN.1 struktury v souladu se specifikací PKCS#8.
d) Zašifruje DH klíčový pár pomocí klíče zprávy analogicky, jako je popsáno v článku 3.5.6.
e) Opatří jej HMAC podpisem, analogicky, jako je popsáno v článku 3.5.7.1.
f) Odešle prostřednictvím serveru na NZ.

NZ zkontroluje integritu a autenticitu zprávy ověřením platnosti platnost HMAC podpisu a v pozitivním případě provede následující kroky:

a) Rozšifruje DH klíčový pár analogicky, jako je popsáno v článku 3.5.8.2.
b) Zkontroluje shodu GUID s hodnotou, kterou obdržel v chybové zprávě serveru při pokusu o registraci.
c) Zkontroluje shodu veřejné části DH klíčového páru s hodnotou, která byla použita pro výpočet autentizačního PIN.

V případě, kdy všechny kontroly proběhnou úspěšně, NZ smaže svůj vygenerovaný DH klíčový pár a nahradí jej klíčovým párem získaným popsaným postupem ze SZ.

NZ pošle serveru novou žádost o registraci, tentokrát s DH klíčovým párem přeneseným ze SZ.

3.5.11.2 PŘEPSÁNÍ PŮVODNÍCH KLÍČŮ NOVÝMI

Pokud uživatel zvolí možnost přepsat původní klíče novými, server odstraní všechna dříve registrovaná zařízení uživatele a revokuje původní klíč.

3.6 APLIKAČNÍ ŠIFROVÁNÍ DAT ULOŽENÝCH NA ZAŘÍZENÍ

Na zařízeních jsou uloženy šifrované zprávy a přiložené dokumenty, zašifrované klíče zpráv pro šifrování/dešifrování zpráv a dokumentů (Message key) a konečně zašifrované klíče zařízení (Device key), které šifrují/dešifrují klíče zpráv a privátní část DH klíčového páru.

3.6.1 DATABÁZE SQLite

Veškeré konverzace (odeslané a přijaté zprávy) jsou na zařízení uloženy v databázi SQLite. Všechny položky jsou do databáze ukládány v zašifrovaném tvaru, jak je popsáno dále v 3.6.9. Všechny dokumenty (odeslané a stažené přílohy) jsou na zařízení uloženy v zašifrovaných

souborech v souborovém systému sandboxu aplikace BabelApp, jak je popsáno dále v 3.6.9.

3.6.2 HESLO UŽIVATELE

Základem aplikační bezpečnosti klienta BabelApp je kvalitní heslo uživatele, pro jeho volbu platí doporučení uvedené v článku 5.1.

Heslo uživatele je využíváno pro 2 účely:

- odvození klíče pro šifrování a dešifrování klíče zařízení DK
- autentizace uživatele k BabelApp klientu na zařízení

Uživatelské heslo je nutno zadat v následujících případech:

Windows PC nebo macOS:

- při přihlášení k aplikaci po spuštění, nebo uzamčení aplikace

Mobilní zařízení s iOS a Android:

- při spuštění klienta BabelApp, tedy po restartu zařízení, nebo pokud systém odstranil klienta BabelApp z paměti
- pokud byl u spuštěné aplikace opakovaně zadán neplatný PIN, nebo neplatný otisk prstu

3.6.3 KLÍČ ZAŘÍZENÍ

Klíč zařízení DK je na každém zařízení uživatele různý, vygenerovaný jako náhodný 256 bitový AES klíč:

DK = RND(32)

Pokud není klient BabelApp spuštěn, zůstává klíč zařízení zašifrovaný (eDK) pomocí 256 bitového AES klíče PK, odvozeného z hesla uživatele a náhodně vygenerované hodnoty soli:

PK = PBKDF2[hmacWithSHA256](heslo_uživatele, sůl, počet_iterací, 256)

eDK = ENC_ECB[PK](DK)

Po spuštění klienta BabelApp a zadání hesla se spočítá klíč PK, s kterým se rozšifruje klíč zařízení DK:

DK= DEC-ECB[PK](eDK)

Rozšifrovaný klíč DK zůstává v paměti po celou dobu, pokud je aplikace spuštěna a to i když je na pozadí.

3.6.4 AUTENTIZACE UŽIVATELE KE KLIENTU BABELAPP

Autentizace uživatele ke klientu BabelApp je založena na zadání hesla uživatele a porovnání vypočtené a na zařízení uložené hodnoty KCV, kryptograficky odvozené z klíče zařízení DK.

Klíč DK musel být nejprve rozšifrován, jak je popsáno v 3.6.3. Hodnota KCV je spočtena zašifrováním konstanty pomocí AES šifry s klíčem DK v ECB módu:

KCV = ENC_ECB[DK](konstanta)

Při shodě uložené a vypočtené hodnoty KCV je uživatel autentizován a aplikace se odemkne.

3.6.5 DH KLÍČE UŽIVATELE

Každý uživatel vlastní DH klíčový pár – privátní klíč X, který je tajný a korespondující veřejný klíč Y, který je registrován v rámci účtu uživatele na serveru. DH klíče jsou generovány na zařízení uživatele a případně přenášeny mezi zařízeními, jak je popsáno v 3.5.11.

Privátní klíč X je na zařízení uživatele uložen v zašifrovaném tvaru. Pro šifrování klíče X se používá algoritmus AES v módu ECB a klíč zařízení DK:

eX = ENC-ECB[DK](X)

Privátní klíč X se používá pro nalezení shody na klíči kontaktu CK.

3.6.6 KLÍČE KONTAKTŮ

Klíče kontaktů CK (a rovněž klíče pro kontrolu integrity) jsou vypočteny na základě veřejného DH klíče kontaktu a privátního DH klíče uživatele postupem uvedeným v 3.5.2.

Klíče kontaktů jsou uloženy na zařízení uživatele v zašifrovaném tvaru. Pro šifrování klíče CK se používá algoritmus AES v módu ECB a klíč zařízení DK:

eCK = ENC-ECB[DK](CK)

Klíč kontaktu CK se používái pro šifrování a dešifrování klíčů zpráv MK, kterými jsou zašifrovány konverzace s daným kontaktem.

3.6.7 ODEMČENÍ MOBILNÍHO KLIENTA BABELAPP POMOCÍ PIN

Po autentizaci uživatele k mobilnímu klientu zadáním hesla uživatele je rozšifrován klíč DK a ponechán v paměti, dokud je aplikace spuštěna. Z bezpečnostních důvodů je vhodné aktivovat automatické zamykání aplikace po uplynutí nastavené doby nečinnosti, nebo kdykoli zamknout aplikaci manuálně.

Na mobilním zařízení není praktické zadávat silné heslo uživatele (viz 5.1) při každém odemčení klienta BabelApp. Z tohoto důvodu lze aktivovat zamykání a odemykání klienta v autentizovaném stavu (s rozšifrovaným klíčem zařízení DK) pomocí zadání 4 místného číselného kódu PIN.

Kód PIN lze na mobilním zařízení nastavit a měnit po zadání hesla uživatele.

Odemčení klienta je založeno na zadání PIN a porovnání vypočtené a na zařízení uložené hodnoty PCV, kryptograficky odvozené z hodnoty PIN a náhodně zvolené

hodnoty soli pomocí funkce PBKDF2:

PCV = PBKDF2[hmacWithSHA1]
(PIN, sůl, počet_iterací, 128)

Při shodě uložené a vypočtené hodnoty PCV se aplikace odemkne.

Poznámka:
Klienti pro počítač s macOS a Windows nemají implementován mechanismus PIN, místo toho je vždy použito zadání silného hesla. Na plnohodnotné klávesnici nečiní zadání silného hesla problém.

3.6.8 ODEMČENÍ MOBILNÍHO BABELAPP KLIENTA POMOCÍ OTISKU PRSTU

Pokud systém mobilního zařízení podporuje autentizaci pomocí otisku prstu, je možné nastavit odemykání klienta BabelApp pomocí otisku prstu jako alternativu k odemykání pomocí kódu PIN.

3.6.9 ŠIFROVÁNÍ ZPRÁV A PŘÍLOH ULOŽENÝCH NA ZAŘÍZENÍ

Při transportu je každá zpráva T zašifrována náhodným klíčem MK, klíč MK je zašifrován klíčem kontaktu CK, jak je popsáno v 3.5.6.2.

Před uložením přijaté zprávy na zařízení je nejprve postupem popsaným v 3.5.8.2 rozšifrován klíč zprávy MK:

MK = DEC-ECB[CK](eMK)

Poté je klíč zprávy zašifrován prostřednictvím klíče zařízení:

eMK = ENC-ECB[DK](MK)

a následně je zašifrovaná zpráva a zašifrovaný klíč zprávy eMK uložen na zařízení.

Poznámka:
Jak je z popisu zřejmé, zpráva (i přílohy) zůstávají při transportu i uložení na zařízení šifrovány stejným klíčem MK, při uložení se pouze přešifrovává samotný klíč MK.

3.6.10 DEŠIFROVÁNÍ ZPRÁV A PŘÍLOH ULOŽENÝCH NA ZAŘÍZENÍ

Z hesla uživatele je odvozen AES klíč PK, kterým se rozšifruje klíč zařízení DK, jak je popsáno v 3.6.3. Pomocí klíče DK lze rozšifrovat všechny klíče zpráv eMK:

MK = DEC-ECB[DK](eMK)

a následně rozšifrovat všechny zprávy eT analogickým způsobem, jako je popsáno v 3.5.8.2:

T = DEC-CBC[MK,IV](eT), kde IV=0

a rovněž pomocí MK rozšifrovat metadata příloh a přiložené dokumenty analogickým způsobem jako je popsáno v 3.5.10.

3.7 APLIKAČNÍ ŠIFROVÁNÍ PŘI TELEFONICKÝCH HOVORECH VOIP

BabelApp poskytuje šifrované telefonické hovory mezi mobilními zařízeními v datové síti prostřednictvím sady technologií VoIP - zejména signalizace pro řízení spojení a výměnu veřejných klíčů, kodeků a protokolů pro přímou komunikaci mezi zařízeními se zajištěním důvěrnosti a integrity.

V případě VoIP probíhá přenos hlasu přímo mezi koncovými klienty, pokud to umožňuje způsob připojení klientů do internetu, nebo prostřednictvím TURN serveru (Traversal Using Relays around NAT https://tools.ietf.org/html/rfc5766) v případě, kdy je nutné spojení reléovat.

BabelApp zajišťuje bezpečný přenos hlasu na základě aplikačního šifrování protokolem SRTP (Secure Real Time Protocol, https://tools.ietf.org/html/rfc3711) mezi koncovými body přenosu. Pro použití šifrovacích klíčů je využita technologie Perfect Forward Secrecy.

3.7.1 PERFECT FORWARD SECRECY

Pro nalezení a použití sdílených šifrovacích klíčů je využita technologie Perfect Forward Secrecy, která zaručuje, že v případě kompromitace DH klíčů, nebo klíčů nějaké relace nejsou kompromitovány klíče žádné jiné relace (minulé, ani budoucí).

Volající strana si před každým hovorem vygeneruje nový dočasný DH key-pair a veřejnou část odešle příjemci v signalizační zprávě sess-initiate. Příjemce po přijetí sess-initiate vygeneruje svůj nový dočasný DH key-pair a příslušný veřejný klíč odešle volající straně v signalizační zprávě sess-accept.

Protože jsou před každým hovorem generovány nové dočasné DH klíče volajícího a volaného, musí být zajištěna kontrola jejich autenticity a integrity, aby byl znemožněn MiTM útok. K tomuto účelu je v signalizačních zprávách použit kontrolní mechanismus založený na algoritmu HMAC, popsaný v článku 3.5.7.1. Klíč použitý v algoritmu HMAC pro kontrolu integrity a autenticity signalizačních zpráv je odvozen z dlouhodobých DH klíčů obou stran (viz 3.5.2.2) a umožňuje tudíž kontrolu integrity a autenticity dočasných DH klíčů.

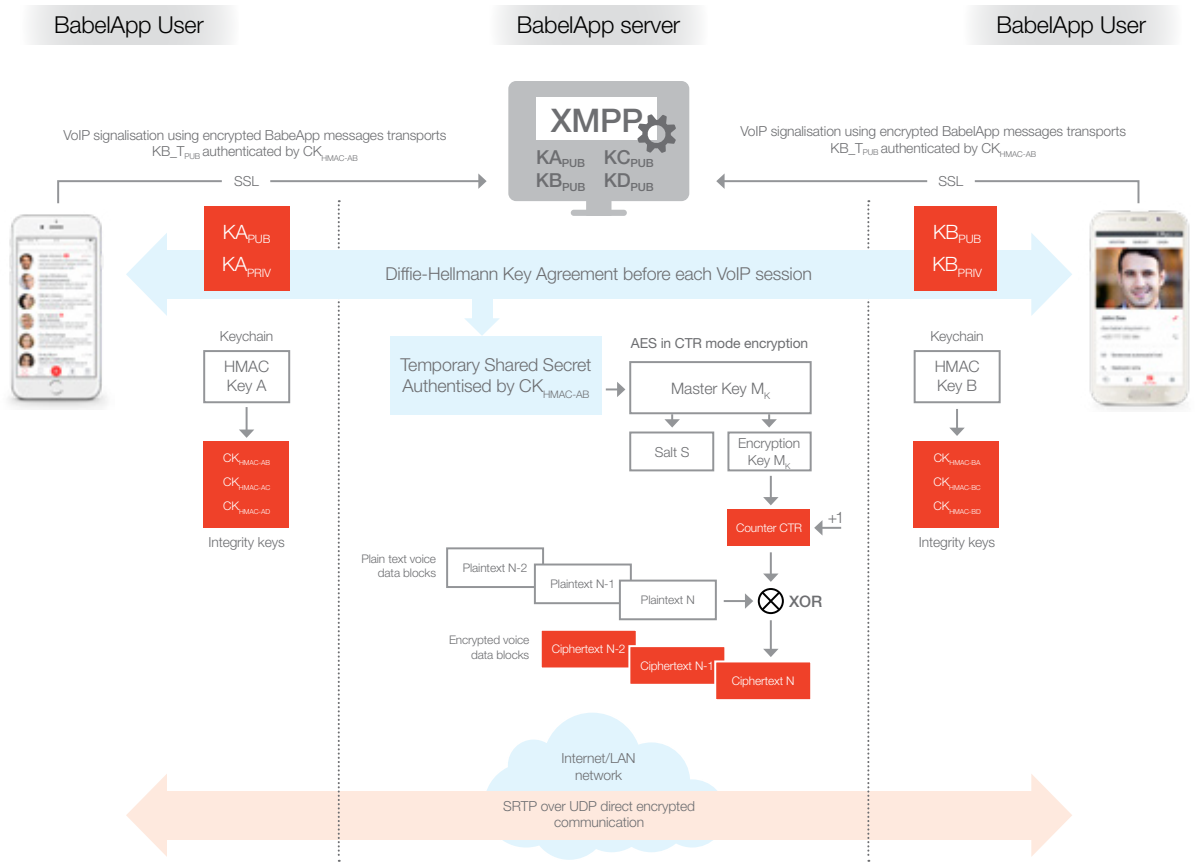
Obě strany spočítají postupem uvedeným v článku 3.5.1.4 dočasné sdílené tajemství ZT, z kterého vygenerují klíč MasterKey MK s délkou 256 bitů postupem uvedeným v 3.5.2.2.

Klíč MK je použit jako vstup do SRTP knihovny, která z něj odvodí AES128 klíč relace a inicializační vektor (sůl) pro šifrování proudu dat RTP.

Všechny klíče jsou umístěny pouze v paměti klientské aplikace, neukládají se a ihned po ukončení hovoru se klíče odstraní.

3.7.2 ROZŠÍŘENÍ KRYPTOGRAFICKÉHO MODELU PRO VOIP

Na rozdíl od asynchronního předávání zpráv prostřednictvím BabelApp serveru vyžaduje VoIP efektivní přímou komunikaci mezi klienty (výjimku tvoří využití TURN serveru pro reléování). Při přímé komunikaci se používá SRTP protokol, přenášející aplikačně zašifrovaný proud datových bloků prostřednictvím UDP transportních paketů. Uvedený transportní mechanismus vyžaduje rozšířený kryptografický model, který využívá stávající bezpečný přenos BabelApp zpráv pro VoIP signalizaci, existující klíče pro kontrolu integrity zpráv k autentizaci dočasných DH klíčů a šifrování algoritmem AES v čítačovém módu pro odolnost při ztrátě paketů.



3.7.3 ŠIFROVÁNÍ PROUDU DAT RTP

V rámci SRTP se RTP proud dat šifruje pomocí algoritmu AES-128 (FIPS PUB 197) v čítačovém módu (CTR mód) podle NIST SP 800-38A, článek 6.5. Šifrování proudu dat se provádí pomocí šifrovacího klíče EK odvozeného ze sdíleného klíče M_K.

Principem čítačového módu je šifrování posloupnosti celých čísel s randomizovanou počáteční hodnotou čítače. Důvodem použití CTR módu je nespolehlivý přenos RTP dat v rámci protokolu UDP, kdy může docházet k výpadku paketů. CTR mód umožňuje šifrovat a dešifrovat jednotlivé bloky při znalosti hodnot čítače nezávisle na sobě a výpadek části zašifrovaného proudu dat neovlivní schopnost příjemce rozšifrovat následující data.

Při použití čítačového módu je kritická počáteční hodnota čítače, která musí být pro každou hodnotu šifrovacího klíče EK různá a samozřejmě známá oběma stranám komunikace. Proto je počáteční hodnota čítače odvozena z kombinace SSRC (náhodně zvolený 32 bitový identifikátor, který je jedinečný v rámci určité RTP relace) a soli, která je odvozena ze sdíleného klíče M_K.

3.7.3.1 ODVOZENÍ ŠIFROVACÍHO KLÍČE (ENCRYPTIONKEY EK)

Pro odvození šifrovacího klíče EK z hodnoty Master klíče MK se použije šifrování AES128 v ECB módu následovně:
Z MK se vezme prvních 16 byte, které budou tvořit hodnotu AES128 klíče K.
Šifrovaná data M1 tvoří následujících 14 byte doplněných zprava nulami na délku 16 byte.

$E_K = ENC\_ECB[K](M)$

Příklad

MK = f6c8fe882700ea330f416e9bf9e970381 cb56a2eb2ed87becdef48150a2d69ea

K = f6c8fe882700ea330f416e9bf9e97038

M1 = 1cb56a2eb2ed87becdef48150a2d0000

E_K = ENC_ECB[f6c8fe882700ea330f416e9bf9e97038] (1cb56a2eb2ed87becdef48150a2d0000) = ddb089593a0c8478670ae70f576780f4

IV_{Name} = RND(16)

IV_{Type} = RND(16)

$IV_{Thumbnail}$ = RND(16)

3.7.3.2 ODVOZENÍ SOLI (S)

Sůl S se vygeneruje podobně jako E_K:

Z M_K se vezme prvních 16 byte, které budou tvořit hodnotu AES128 klíče K.

Šifrovaná data M2 tvoří následujících 14 byte doplněných zprava nulami na délku 16 byte, na které se provede operace XOR s konstantou 00000000000000020000000000000000.

Příklad

Mk = f6c8fe882700ea330f416e9bf9e970381c-b56a2eb2ed87becdef48150a2d69ea

K = f6c8fe882700ea330f416e9bf9e97038

M2 = 1cb56a2eb2ed87becdef48150a2d0000 XOR 000 00000000000020000000000000000 = 1cb56a2eb2e d87bccdef48150a2d0000

S = ENC_ECB[f6c8fe882700ea330f416e9bf9e97038] (1cb56a2eb2ed87bccdef48150a2d0000) =

a8f746c53802b7391dd505113bbc13d7

3.7.3.3 ČÍTAČ (CTR)

Struktura čítače je následující:

- Byte 0 – 3: vyplněno zleva 0Byte 4 – 7: SSRC
- Byte 8 – 13: čítač paketů
- Byte 14 – 15: čítač šifrovaných bloků v paketu

SSRC (Synchronization source) je náhodně zvolená hodnota o délce 4 byte, která musí být jedinečná v rámci RTP relace. Hodnota SSRC je přenášena v hlavičce každého RTP paketu – viz https://tools.ietf.org/html/rfc3550.

Inicializace čítače CTR

Při navázání RTP relace je čítač inicializován následovně:

- Byte 0 – 3: 0000H
- Byte 4 – 7: SSRC
- Byte 8 – 13: 000000H
- Byte 14 – 15: 00H

Počáteční hodnota čítače CTR se získá operací XOR následovně:

CTR = CTR XOR S0

S0 je upravená hodnota soli S, získaná podle 3.7.2.2, kde jsou poslední 2 byte vynulovány.

Příklad

SSRC = 048e3f1d

S = a8f746c53802b7391dd505113bbc13d7

S0 = a8f746c53802b7391dd505113bbc0000

CTR = 00000000048e3f1d0000000000000000 XOR a8f746c53802b7391dd505113bbc0000 = a8f746c53c8c88241dd505113bbc0000

3.7.3.4 ŠIFROVÁNÍ PROUDU DAT

SRTP generuje proud výstupních bloků šifrováním hodnot čítače CTR. Každý blok v proudu výstupních bloků je vytvořen šifrováním hodnoty čítače CTR klíčem EK.
Výsledná zašifrovaná data jsou tvořena operací XOR tohoto výstupního bloku a bloku otevřených dat v paketu.
Pro šifrování následujícího bloku otevřených dat v paketu se inkrementuje hodnota byte 14 – 15 (čítač bloků) v CTR.
Pro šifrování následujícího paketu se inkrementuje hodnota byte 8 – 13 (čítač paketů) a současně vynuluje hodnota byte 14 – 15 (čítač bloků) v CTR.

Příklad

Příklad ukazuje inkrementaci čítače CTR

První blok v prvním paketu

CTR = 00000000048e3f1d0000000000000000 XOR a8f746c53802b7391dd505113bbc0000 = a8f746c53c8c88241dd505113bbc0000

Druhý blok v prvním paketu

CTR = 00000000048e3f1d0000000000000001 XOR a8f746c53802b7391dd505113bbc0000 = a8f746c53c8c88241dd505113bbc0001

První blok v druhém paketu

CTR = 00000000048e3f1d0000000000010000 XOR a8f746c53802b7391dd505113bbc0000 = a8f746c53c8c88241dd505113bbd0000

Příklad

Příklad ukazuje šifrování dat v otevřeném tvaru pomocí výstupních bloků vzniklých šifrováním hodnot čítače CTR

Data v otevřeném tvaru:

M = 0800a0e72b6096e720ad92788189e6fa70395 fd8bb942ee8a54c7ea8

První blok:
Zpráva M se rozdělí na 128 bitové bloky M1 a M2

M1 = 0800a0e72b6096e720ad92788189e6fa

E_K = ddb089593a0c8478670ae70f576780f4

CTR = a8f746c53c8c88241dd505113bbc0000

OB = ENC_ECB[E_K](CTR)

OB1 = ENC_ECB[ddb089593a-0c8478670ae70f576780f4](a8f746c53c8c88241dd 505113bbc0000) = 31e87f4515807246714bb245 efa57e60

Výsledný zašifrovaný 1. blok C1:
C1 = OB1 XOR M1

C1 = 31e87f4515807246714bb245efa57e60 XOR 0800a0e72b6096e720ad92788189e6fa = 39e8dfa23ee0e4a151e6203d6e2c989a

Druhý blok:

M2 = 70395fd8bb942ee8a54c7ea8

EK = ddb089593a0c8478670ae70f576780f4

CTR = a8f746c53c8c88241dd505113bbc0001

OB = ENC_ECB[E_K](CTR)

OB2 = ENC_ECB[ddb089593a0c8478670ae70f576780f4](a8f746c53c8c88241dd505113bbc0001) = c420698fdb98de7715721a50af0955f1

Výsledný zašifrovaný 2. blok C2:
C2 = OB2 XOR M2

C2 = c420698fdb98de7715721a50 XOR 70395fd8 bb942ee8a54c7ea8 = b4193657600cf09fb03e64f8

Zašifrovaný řetězec M = M1 || M2

39e8dfa23ee0e4a151e6203d6e2c989 ab4193657600cf09fb03e64f8

3.7.4 INTEGRITA RCP PAKETU

Každý RTP paket je opatřen kontrolou integrity na základě HMAC s hashovací funkcí SHA-1.

3.8 SYSTÉMOVÉ ŠIFROVÁNÍ

Jádrem bezpečnosti dat uložených na klientu BabelApp je aplikační šifrování, které je nezávislé na operačním systému klienta, v kterém jsou data uložena. Některé operační systémy umožňují k aplikačnímu šifrování přidat další úroveň ochrany prostřednictvím systémového šifrování.

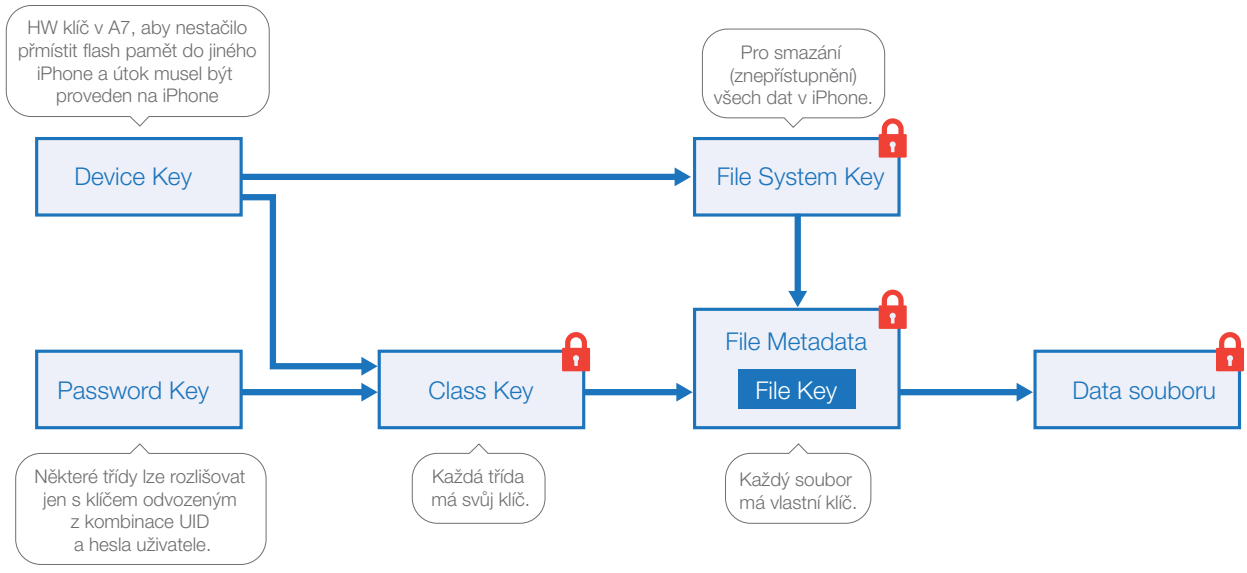
3.8.1 iOS

Všechna data v iOS jsou zašifrována ve flash paměti zařízení prostřednictvím hierarchické struktury klíčů, zobrazené na následujícím schématu.

Podmínkou pro ochranu systému souborů šifrováním je nastavení hesla iOS zařízení. iOS uměle zpomaluje snahy o pokus uhádnutí hesla hrubou silou (cca na 80 ms na jeden pokus). Útok hrubou silou se musí dělat přímo na zařízení, neboť heslo je kombinováno s UID v procesoru.

Úroveň systémové kryptografické ochrany pro zprávy uložené v SQLite databázi a přílohy uložené v souborovém systému sandboxu aplikace BabelApp je nastavena na hodnotu NSFileProtectionCompleteUntilFirstUserAuthentication, která rozšifruje Class Key v okamžiku zadání hesla uživatele iOS zařízení a ponechá jej v paměti zařízení i po uzamčení aplikace.

Dobrý kompromis mezi nepohodlím častého zadávání komplexního hesla při odemykání iOS zařízení a bezpečností lze dosáhnout s využitím biometriku otisku prstů (platí pro iPhone 5s resp iPad 3 a vyšší). Relativně nízké FAR biometrické autentizace je kompenzováno pomocí nutnosti zadat heslo po 5 neúspěšných pokusech s otiskem prstu.



Systémové šifrování iOS

3.8.2 ANDROID

Systémové šifrování není použito.

3.8.3 WINDOWS

Systémové šifrování není použito.

3.8.4 macOS

Systémové šifrování není použito.

4. PLATFORMA SERVERU

BabelApp server vyžaduje pro svůj běh následující prostředky (pro detaily viz Implementation Guide):

4.1 HARDWARE

PC server s CPU Intel x86/x64, taktovaný na alespoň 2 GHz

- Paměť – alespoň 3 GB
- Disk – volný prostor alespoň 10 GB
- Síťová karta - 100 Mbps nebo vyšší

Server může být fyzický, nebo virtuální. Testované virtualizační platformy jsou VMware a Hyper-V.

4.2 OPERAČNÍ SYSTÉM

BabelApp vyžaduje operační systém Microsoft Windows server, nebo Linux:

Linux

- Oracle Linux Server 8.x
- Red Hat Enterprise 8.x
- CentOS/Almalinux 8.x

4.3 JAVA

BabelnApp server vyžaduje instalaci 32 bit Java 1.8 a vyšší.

4.4 DATABÁZE

BabelApp server používá databázový systém PostgreSQL, minimální verze 9.6, preferovaná verze 13.

4.4.1 Účet v operačním systému, pod kterým běží PostgreSQL

Linux: Server PostgreSQL musí být spouštěn pod vyhrazeným unix účtem se silným heslem, nikoli pod účtem root, nebo jiným uživatelským účtem. Tento vyhrazený účet by měl vlastnit pouze data, která jsou spravována serverem a nesmí být sdílen s jinými službami.

Windows: Služba server PostgreSQL musí být spouštěna pod vyhrazeným účtem, který má nastavena práva pouze k datům, které spravuje. Tento vyhrazený účet musí mít právo read na všechny adresáře tvořící cestu k adresáři služby a právo write pouze na složku data.

4.4.2 Účet superuživatele

Při instalaci je nutné změnit implicitní jméno a heslo superuživatele PostgreSQL (postgres:postgres) na silné heslo (příkaz ALTER ROLE).

```
ALTER ROLE postgres WITH PASSWORD ‘some_strong_password’
```

4.4.3 Založení databází pro BabelApp

Při instalaci je nutné založit 3 databáze:

- openfire
- babel
- babel_attachment

4.4.4 Vytvoření databázových účtů

K uvedeným databázím je nutné vytvořit databázové účty (uživatele s rolí LOGIN privilege k příslušné databázi) a nastavit jim silná hesla. Databázové účty se pojmenují stejně, jako jména databází:

```
CREATE USER openfire WITH PASSWORD strong_password_1'
CREATE USER babel WITH PASSWORD strong_password_2'
CREATE USER babel_attachment WITH PASSWORD strong_password_2'
```

4.4.5 Nastavení způsobu autentizace pro databázové účty

PostgreSQL podporuje několik autentizačních metod, které jsou určeny souborem pg_hba.conf, umístěným v kořenovém adresáři databázového serveru.

Doporučuje se omezit připojení pouze na lokální připojení a metodu autentizace nastavit na autentizaci vůči lokálnímu operačnímu systému serveru (peer).

Peer Authentication (pouze pro lokální autentizaci)
Pokud je BabelApp server instalován na stejném HW (nebo na stejném virtuálním serveru) jako PostgreSQL, lze využít autentizaci vůči lokálním účtům operačního systému. Jedná se o preferovanou volbu ***peer***.

Databázové účty jsou logicky oddělené od uživatelských účtů v operačním systému, pokud ale vytvoříme stejné účty v operačním systému jako v PostgreSQL, lze využít autentizační metodu peer authentication pro lokální připojení.

V postgress.conf nastavit
listen_addresses na prázdný seznam adres

Struktura pg_hba.conf pro lokální připojení:

# TYPE	DATABASE	USER	METHOD
local	sameuser	all	peer

Password Authentication (pro síťovou autentizaci)

Password authentication s volbou md5 provádí autentizaci klienta prostřednictvím hesla, které je na klientu dvakrát hashováno, jednou po zřetězení se jménem uživatele a podruhé se solí.

V postgress.conf nastavit

listen_addresses na povolenou síťovou adresu klienta, například (IPv4) 192.168.1.0/24
port na TCP port, implicitně 5432

Struktura pg_hba.conf pro síťové připojení:

Předpokládá se, že klient (BabelApp server) je umístěn na adrese 192.168.1.0/24

# TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
host	openfire	openfire	192.168.1.0/24	md5
host	babel	babel	192.168.1.0/24	md5
host	babel_attachment	babel_attachment	192.168.1.0/24	md5

4.5 OPENFIRE

BabelApp server používá XMPP server Openfire http://www.igniterealtime.org/projects/openfire/, který je implementován v prostředí java a licencován pod Open Source Apache License. Openfire umožňuje rozšíření svých funkcí prostřednictvím zásuvných modulů (plugins), funkce BabelApp jsou realizovány prostřednictvím zásuvného modulu babel.jar.

5. BEZPEČNOSTNÍ PŘEDPOKLADY A DOPORUČENÍ

5.1 SILNÉ HESLO

Za silné heslo považujeme heslo s minimální entropií 70 bitů.

Následující tabulka ukazuje entropii znaku hesla, pokud je znak náhodně vybrán z podmnožiny stejně pravděpodobných znaků:

- číslice (10 znaků), entropie 3,32 bitů/znak
- malá písmena bez diakritiky (26 znaků), entropie 4,7 bitů/znak
- velká písmena bez diakritiky (26 znaků), entropie 4,7 bitů/znak
- malá i velká písmena bez diakritiky (52 znaků), entropie 5,7 bitů/znak
- malá i velká písmena bez diakritiky a číslice (62 znaků), entropie 5,95 bitů/znak

Příkladem takového hesla je heslo, které se skládá z náhodné kombinace malých a velkých písmen a číslic, s minimální délkou 12 znaků.

5.2 iOS

Pro mobilní zařízení s iOS platí následující bezpečnostní předpoklady a doporučení:

Předpoklady
- minimální verze iOS je 7.0
- zařízení nemá aplikován jailbreak
- zařízení má aktivován kódový zámek
- BabelApp má nastaveno silné heslo

Doporučení
- aktivovat i smazání dat zařízení po 10 neúspěšných pokusech o zadání kódu.
- používat Touch-ID/Face-ID pro odemknutí zařízení a současně nastavit silné heslo pro kódový zámek
- heslo BabelApp je různé od hesla kódového zámku
- nastavit Použití Touch-ID/Face-ID pro přihlášení k BabelApp

5.3 ANDROID

Pro mobilní zařízení se systémem Android platí následující bezpečnostní předpoklady a doporučení:

Předpoklady
- minimální verze Android je 8
- zařízení nemá aplikován root

Doporučení
- nastavit šifrování dat v systému (minimální verze Android je 11)
- zařízení má aktivován číselný PIN
- nepoužívat gesta k zadání PIN
- aktivovat smazání dat na dálku službou Google
- používat otisk prstu pro odemknutí zařízení
- heslo BabelApp je různé od hesla kódového zámku

5.4 WINDOWS

Pro PC/notebook s Windows platí následující bezpečnostní předpoklady a doporučení:

Předpoklady
- minimální verze Windows je Vista
- zařízení neobsahuje škodlivý kód a je chráněno proti napadení škodlivým kódem
- uživatel má nastaveno silné heslo
- BabelApp má nastaveno silné heslo

Doporučení
- heslo BabelApp je různé od hesla uživatele Windows
- minimální verze Windows je 7

5.5 macOS

Pro Mac s OS platí následující bezpečnostní předpoklady a doporučení:

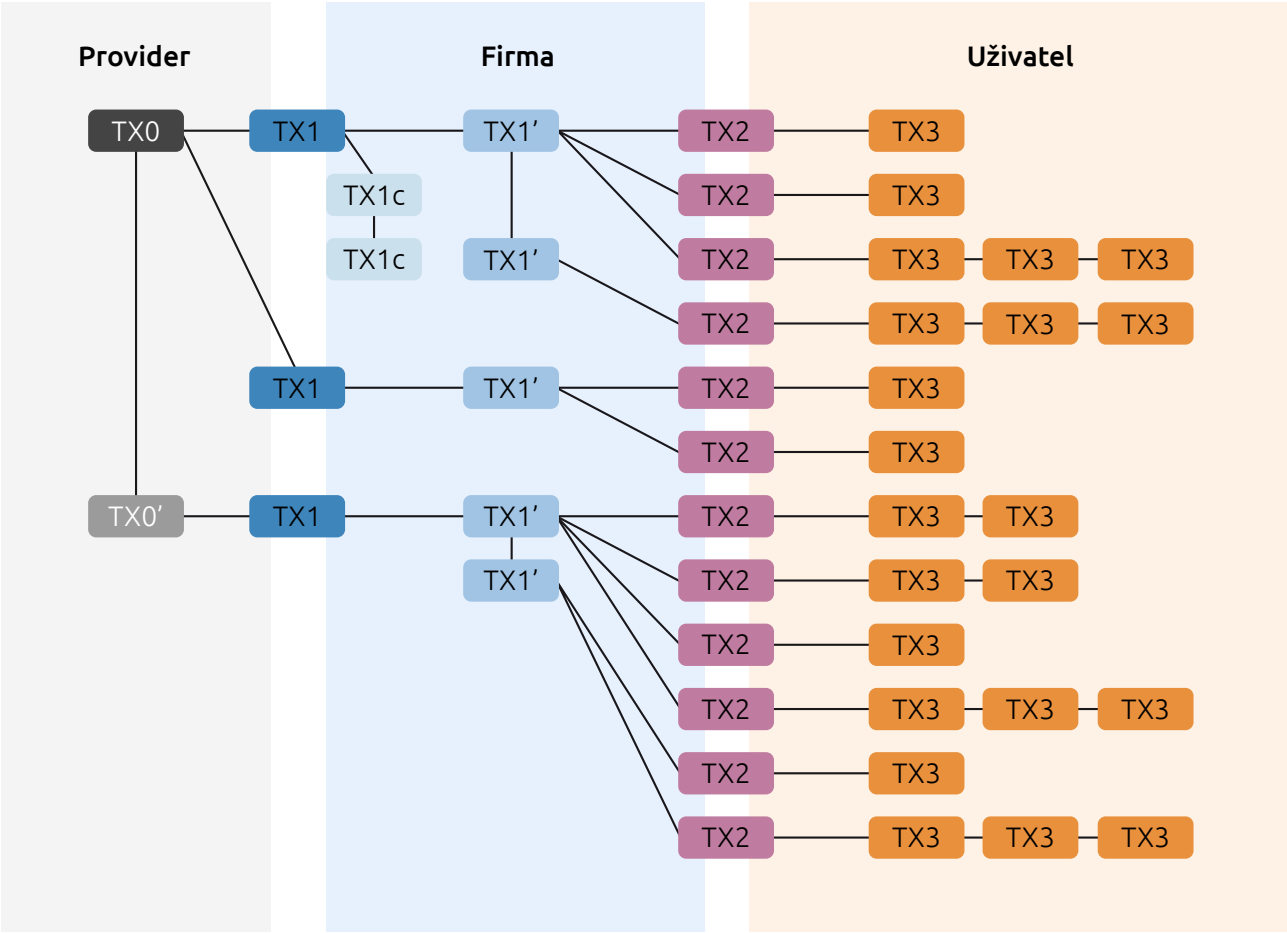
Předpoklady
- minimální verze macOS je 10.11
- zařízení neobsahuje škodlivý kód a je chráněno proti napadení škodlivým kódem
- uživatel má nastaveno silné heslo
- BabelApp má nastaveno silné heslo

Doporučení
- heslo BabelApp je různé od hesla systému

6. OCHRANA POMOCÍ BITCOINOVÉ SÍTĚ

Při aktivované ochraně pomocí Bitcoinové sítě dojde k vytvoření jedné nebo víceBitcoinových transakci, které jsou následně odeslány do Bitcoinové sítě, kde jsou ověřeny a zařazeny do bloku transakcí. Takto zapsané transakce je možné kýmkoli číst, ale již je velice obtížné (prakticky nemožné) takovéto transakce změnit, nebo odstranit.

Transakce v Bitcoinové síti vždy pouze převádějí částku z jedné, nebo více předchozích transakcí na transakci novou. Zda lze částku převést zajišťují elektronické podpisy. Tímto způsobem lze z Bitcoinových transakcí sestavit stromovou strukturu transakcí. Kořenem tohoto stromu je transakce TX0, která je vytvořená firmou OKsystem a.s. (dále jen providerem) a která musí být předkem jakékoli jiné transakce patřící do BabelApp ochrany. Naopak na listech tohoto stromu jsou transakce obsahující klientské údaje pro ověření klíčů. Tyto klientské údaje již vytvářejí a podepisují přímo klienti (uživatelské aplikace BabelApp na telefonech, nebo desktopech) a nikdo jiný je vytvořit nemůže.



6.1 TRANSAKCE

Všechny Bitcoinové transakce použité pro BabelApp ochranu mají stejnou strukturu.

1. vstup odkazuje vždy na předchozí transakci, která je součástí ochrany. Toto platí až k TX0, která je první transakcí BabelApp ochrany a žádného předka tedy již nemá. Její vstup je pouze peněženka pro placení nákladů za TX0 transakci.

2. vstup není povinný a pokud existuje je to vždy peněženka pro placení nákladů za transakci.

1. výstup obsahuje vždy data použitá pro BabelApp ochranu. Výstup je realizován skriptem OP_RETURN, neobsahuje žádnou částku a není možné k němu nikdy žádnou další transakci připojit.

Data vždy začínají dvěma byty, které určují typ transakce.

- bb0a - TX0
- bb0b - TX0'
- bb1a - TX1
- bb1b - TX1'
- bb1c - TX1_C
- bb2a - TX2
- bb3a - TX3

Například: RETURN PUSHDATA(2)[bb1b]

Podle typu transakce může skript obsahovat další informace.

Například: RETURN PUSHDATA(2)[bb1a] PUSHDATA(10)[54819c66332fa033f75c] PUSHDATA(10)[ca62a9103415ebe85fdd]

Další výstupy, které transakce obsahuje jsou závislé na jejím typu.

6.1.1 Kořenová (root) transakce - TX0

Při spuštění ochrany pomocí Bitcoinové sítě byla vytvořena transakce TX0. Transakce byla 19.12.2018 zařazena do bloku 000000000000000000177b0bf7514b0c08e49dff00184362f2104a8820428d65 a její hash je 683db0ae401581952a80b9672fb3b0abc404a558d2681ec4490a78231c9f762f

Data: [bb0a]

Vstupy:
1. vstup: peněženka

Výstupy:
1. výstup: data (bb0a)
2.-11. výstup: P2PK pro připojení TX1
12. výstup: P2PK pro připojení TX0'
13. výstup: P2PKH pro zbytek utracené částky.

Hash TX0 i číslo bloku ve kterém je umístěna obsahují všichni klienti BabelApp. Při ověřování stromu transakcí vždy musí skončit u této TX0. Žádné transakce patřící k ochraně BabelApp se nehledají ve starších blocích, než je blok obsahující TX0.

6.1.2 Rozšiřující kořenová (root) transakce TX0'

Pokud budou všechny výstupy, ke kterým je možné připojit transakci TX1 (buď v TX0, nebo v TX0') již obsazené (výstup je utracen), vznikne nová transakce TX0'. Tato transakce bude opět obsahovat několik výstupů pro připojení TX1, jeden výstup pro připojení TX0' a volitelný výstup pro zbytek utracené částky za nově vzniklou TX0' a poplatky pro těžaře za její vložení do bloku v Bitcoinové síti.

Při validaci, zda transakce patří do bezpečnostního mechanismu BabelApp sítě je nutné ověřit, že 1. vstup odkazuje přímo na TX0, popřípadě na TX0' (pak je nutné ve validaci pokračovat až k TX0).

Data: [bb0b]

Vstupy:

1. vstup: TX0, nebo TX0'
2. vstup: peněženka

Výstupy:

1. výstup: data (bb0b)
2 - x. výstup: P2PK pro připojení TX1
x + 1. výstup: P2PK pro připojení TX0'
x + 2. výstup: P2PKH volitelný výstup se zbytkem utracené částky

6.1.3 Firemní transakce TX1

Transakci vytváří provider na základě žádosti společnosti, která vlastní BabelApp server a chce Bitcoinovou ochranu využívat. Touto transakcí se předává vlastníkovi výlučná možnost zakládat pod danou transakci TX1 jakékoli množství dalších transakcí. Provider k TX1 již nemá přístup.

Data: [bb1a] [hash certifikátu] [zřetězené hashe domén (1-6)]

- hash SSL certifikátu: prvních 10 bytů SHA1 thumbprintu
- hashe domén: prvních 10 bytů z hashe HASH160 (SHA256 následovaná RIPEMD160) UTF8 zakódovaného doménového jména

Příklad:

TX1 s jedním doménovým jménem:

RETURN PUSHDATA(2)[bb1a] PUSHDATA(10)[54819c66332fa033f75c] PUSHDATA(10)[ca62a9103415ebe85fdd]

TX1 se 2 doménovými jmény:

RETURN PUSHDATA(2)[bb1a] PUSHDATA(10)[cdadf3bae8ea0174c50a] PUSHDATA(20)[4f855e3cd1ae75b7de98129ac10b041c4ab209de

Vstupy:

1. vstup: TX0, popřípadě TX0'
2. vstup: peněženka

Výstupy:

1. výstup data (bb1a)
2. výstup P2PKH pro připojení TX1', klíč vlastní firma
3. výstup P2SH pro připojení TX1_C pro změnu domény
4. výstup P2PKH pro zbytek utracené částky

6.1.4 Rozšiřující firemní transakce TX1'

Transakci vytváří firma vlastnící BabelApp server. Transakce rozšiřuje množství výstupů pro připojení transakcí s informacemi o jednotlivých kontaktech.

Data: [bb1b]

Příklad: RETURN PUSHDATA(2)[bb1b]

Vstupy:

1. vstup: TX1, popřípadě TX1'
2. vstup: peněženka

Výstupy:

1. výstup: data (bb1b)
2 - X výstup: P2PK pro připojení TX2, popřípadě další TX1'
X + 1. výstup P2PKH pro zbytek utracené částky

6.1.5 Transakce pro změnu domény TX1_C

Transakci se připojuje buď k TX1, nebo při opakované změně domény k předchozí TX1_C. Skript pro připojení je P2SH (MULTISIG). Je potřeba aby tato transakce byla podepsána jak vlastníkem BabelApp serveru, tak i providerem.

Data: [bb1c] [hash certifikátu] [zřetězené hashe domén (1-6)]

- hash SSL certifikátu: prvních 10 bytů SHA1 thumbprintu
- hashe domén: prvních 10 bytů z hashe HASH160 (SHA256 následovaná RIPEMD160) UTF8 zakódovaného doménového jména

Příklad:

TX1_C s jedním doménovým jménem:

RETURN PUSHDATA(2)[bb1a] PUSHDATA(10) [54819c66332fa033f75c] PUSHDATA(10) [ca62a9103415ebe85fdd]

TX1_C se 2 doménovými jmény:

RETURN PUSHDATA(2)[bb1a] PUSHDATA(10) [cdadf3bae8ea0174c50a] PUSHDATA(20) [4f855e3cd1ae75b7de98129ac10b041c4ab209de

Vstupy:

1. vstup: TX1, popřípadě TX1'
2. vstup: peněženka

Výstupy:

1. výstup data (bb1c)
2. výstup P2SH pro připojení další TX1_C pro další změnu domény
3. výstup P2PKH pro zbytek utracené částky

6.1.6 Transakce pro převod na uživatele TX2

Transakci TX2 vytváří vlastník BabelApp serveru a je připojena k TX1'. Touto transakcí se dává možnost vytvářet transakce již konkrétnímu uživateli.

Data: [bb2a]

- hash SSL certifikátu: prvních 10 bytů SHA1 thumbprintu
- hashe domén: prvních 10 bytů z hashe HASH160 (SHA256 následovaná RIPEMD160) UTF8 zakódovaného doménového jména

Příklad:

RETURN PUSHDATA(2)[bb2a]

Vstupy:

1. vstup: TX1'
2. vstup: volitelně peněženka

Výstupy:

1. výstup: data (bb2a)
2. výstup: MULTISIG (vlastník + uživatel) pro připojení TX3

6.1.7 Uživatelská transakce TX3

Transakce TX3 je podepsána jak uživatelem, tak i společností vlastnící server. TX3 obsahuje informace o BabelApp adresách a veřejných klíčích používaných pro šifrování.

Data: [bb3a] [hash DH public klíče, nebo prázdné pole] [0 až 3 pole po 20 bytech - hashe prefixu babel adres]

- Hashovací funkce je HASH160 (tj. SHA256 následovaná RIPEMD160).
- DH veřejný klíč se na bajty převádí bez případné úvodní nuly.
- U BabelApp adres se nejprve prefix zakóduje pomocí UTF-8.

Příklady:

TX3 klíč + 1 babelapp adresa

RETURN PUSHDATA(2)[bb3a] PUSHDATA(20) [11d953bea4fd10d9f0b3238e802f89b7fb7a84e7] PUSHDATA(20)[ec2d242427dfac4704043aaf8cffa03dbf791a3a]

pouze jedna babelapp adresa (bez informace o klíči)

RETURN PUSHDATA(2)[bb3a] 0[] PUSHDATA(20) [14b43a06e4d849dbaa7d836c5e34e2f445c116f4]

Vstupy:

1. vstup: TX2, popřípadě předchozí TX3
2. vstup: volitelně peněženka

Výstupy:

1. výstup: data (bb3a)
2. výstup: MULTISIG (vlastník + uživatel) pro připojení TX3, popřípadě P2SH obsahující redeem skript typu FRIENDS.

6.2 PENĚŽENKA, PLATBY, ČÁSTKY

Každá Bitcoinová transakce znamená převod částky, což je součet ze všech vstupů, na jednotlivé výstupy. Pokud částka převáděná z 1. vstupu nestačí na všechny definované výstupy + poplatky těžařům, je potřeba připojit ještě 2. vstup a částku navýšit. Pokud je částka na vstupech příliš vysoká je možné přidat poslední výstup, který vrací přebytečnou částku do peněženky.

Transakce TX0, TX0' a TX1 vytváří a platí provider. Transakce TX1', TX1_C a TX2 vytváří a platí společnost vlastnící BabelApp server prostřednictvím serverové peněženky. Transakce TX3 vytváří klient, ale server TX3 kontroluje, připodepisuje ji a pokud je potřeba přidává druhý vstup s požadovanou částkou. TX3 nikdy neobsahuje výstup s přeplatkem do peněženky.

6.3 PROVIDER (OKsystem a.s.)

Provider vytvořil TX0 a podle potřeby vytváří rozšiřující TX0'. Pokud se chce jakýkoli BabelApp server připojit do Bitcoinové sítě požádá providera a ten mu na základě certifikátu, který provider již dříve podepsal a všech doménových jmen vytvoří novou TX1. Žádná doménová jména se nesmějí nikdy opakovat, stejně tak i certifikát.

6.4 VLASTNÍK BABELAPP SERVERU

6.4.1 Připojení k Bitcoinové síti

Při aktivování ochrany pomocí Bitcoinové sítě server nejprve načítá, validuje a ukládá bloky z Bitcoinové sítě do DB. Nestahují se veškerá data, která Bitcoinová síť obsahuje (několik stovek GB), ale stačí načíst data od bloku 000000000000000000177b0bf7514b0c08e49dff-00184362f2104a8820428d65, který musí obsahovat TX0.

Při získávání dat z Bitcoinové sítě je nastaven bloom filter, díky kterému je na BabelApp server zaslán již pouze zlomek všech dat. Bloom filter obsahuje identifikátory všech typů transakcí (bba0 - bb3a) spolu s příznakem BLOOM_UPDATE_P2PUBKEY_ONLY. Takto získaná data BabelApp server dále poskytuje všem klientům. BabelApp server má tedy uložené spolu s bloky všechny BabelApp transakce a to jak transakce, které se týkají daného serveru, tak i všech ostatních BabelApp serverů.

6.4.2 Aktivace ochrany pomocí Bitcoinové sítě - vznik TX1

Pro využívání Bitcoinové ochrany, provider nejprve vytvoří transakce TX1. TX1 obsahuje hash aktuálního certifikátu a hashe doménových jmen (common name, alternative name). Druhý výstup pro TX1' je zabezpečen pomocí skriptu P2PKH s adresou vlastníka BabelApp serveru a třetí výstup pro TX1_C je zabezpečen pomocí P2SH s MULTISIG redeem skriptem (adresa providera a vlastníka BabelApp serveru). Takto vytvořenou transakci provider vloží do Bitcoinové sítě a vlastník serveru k ní může připojit TX1' a následně jednotlivé klienty.

Ověření TX1

Jakmile server získá z Bitcoinové sítě TX1 transakci, ověří, že je vytvořena správně

a. ověří, že 2. výstup u TX1 je chráněn P2PKH skriptem s adresou vlastníka BabelApp serveru
b. ověří, že 3. výstup u TX1 je chráněn P2SH skriptem, kde použitý redeem script je MULTISIG 2 ze 2 a jedna z adres patří danému serveru

6.4.3 Změna domény

Transakce TX1, popřípadě TX1_C obsahuje hash aktuálního certifikátu a hashe doménových jmen (common name, alternative name). Při změně domény, vznikne nový certifikát, který obsahuje nové doménové jméno (common name) a staré doménové jméno je v alternative name. Provider založí transakci TX1_C, která obsahuje hash nového certifikátu a hash nové domény. TX1_C se podepisuje providerem i vlastníkem BabelApp serveru. Změna domény BabelApp serveru se tedy provádí pouze za součinnosti jak vlastníka BabelApp serveru, tak i poskytovatele služby.

6.4.4 Ztráta klíče

Pokud dojde ke ztrátě klíče chránícího TX1 (TX1'), je potřeba zažádat o vytvoření nové TX1 s jiným certifikátem a s novými doménami. Všechny kontakty je potom nutné přeregistrovat pod novou doménu. Je proto velice důležité firemní klíče bezpečně uschovat a zálohovat.

6.5 KLIENTI

6.5.1 Validace transakcí

Klienti získávají jednotlivé bloky obsahující transakce ze serveru ke kterému jsou připojeni. BabelApp server poskytuje hlavičky všech bloků (od TX0) a vybrané transakce ve struktuře merkle tree. Takto získaná data klienti ověří, získají z nich BabelApp transakce a uloží je do DB. Data ve struktuře merkle tree není možné měnit, ani žádná přidat, ale je možné data zamlčet. Proto klienti kontaktují více serverů (i babelapp.com) pro informaci ohledně počtu transakci k danému bloku dat. Server zasílá pouze bloky, které mají alespoň 10 potvrzení.

PREPROCESSING PŘI PŘÍJMU TRANSAKCE

a. Bitcoinové validace - všechny bloky i v nich obsažené transakce musejí splňovat pravidla definovaná Bitcoinovou sítí a podle toho se také validují.
b. BabelApp validace - nejprve se parsuje transakce, ověří se, že je to BabelApp transakce a zjistí se její typ.
    i. TX0 - musí se nalézat v bloku 000000000000000000177b0bf7514b0c08e49dff-00184362f2104a8820428d65 a její hash musí být 683db0ae401581952a80b9672fb3b0abc404a-558d2681ec4490a78231c9f762f
    ii. TX0' - musí odkazovat na TX0, nebo již ověřenou TX0'
    iii. TX1 - musí odkazovat na TX0, nebo již ověřenou TX0'
        1. hash certifikátu je unikátní, neobsahuje ho žádná starší ověřená TX1, nebo ověřená TX1_C
        2. hash všech domén je unikátní, žádnou z domén neobsahuje žádná starší ověřená TX1, nebo ověřená TX1_C
    iv. TX1' - musí odkazovat na již ověřenou TX1, nebo již ověřenou TX1'
    v. TX1_C - musí odkazovat na již ověřenou TX1, nebo již ověřenou TX1_C
        1. hash certifikátu je unikátní, neobsahuje ho žádná starší ověřená TX1, nebo ověřená TX1_C
        2. hash všech domén je unikátní, žádnou z domén neobsahuje žádná starší ověřená TX1, nebo ověřená TX1_C, jinak je BabelApp server (TX1 podstrom) označen jako nedůvěryhodný
    vi. TX2 - musí odkazovat na již ověřenou TX1'
    vii. TX3 - musí odkazovat na již ověřenou TX2, nebo již ověřenou TX3
        1. hash žádné z prefixu babelapp adres se již nenachází ve stejném TX1 podstromě (BabelApp server), jinak je BabelApp server označen jako nedůvěryhodný

VALIDACE CIZÍHO UŽIVATELE

Při validaci kontaktu se postupuje v daném pořadí.

a. nalezení a validace serveru (TX1)
    i. podle všech BabelApp adres validovaného kontaktu se naleznou všechny domény
    ii. pomocí SRV se získají certifikáty všech domén a ověří se, že jsou totožné
    iii. z certifikátu se vypočte jeho hash a vyhledá se správný podstrom (TX1)
    iv. ověří se, že certifikát není revokovaný, ani že BabelApp server není nedůvěryhodný
    v. ověří se, že hashe všech domén jsou uloženy u TX1, popřípadě u TX1_C
b. nalezení a validace uživatele (TX3)
    i. podle hashe prefixu BabelApp adresy se v TX1 podstromu naleznou informace o uživateli (TX3 transakce)
    ii. ověří se, že poslední hash DH klíče v TX3 je stejný jako hash DH klíče, který poskytuje BabelApp server
    iii. ověří se, že všechny prefixy BabelApp adres jsou uloženy v daných TX3 transakcích a že žádná nechybí, ani nepřebývá
    iv. pokud BabelApp server poskytl novější klíč, než je v poslední TX3 (bod ii), popřípadě některá z adres chybí v daných TX3 (bod iii), je možné, že se nejedná o útok, ale že ještě nedošlo k vložení transakce s novou změnou do bloku v Bitcoinové síti (popřípadě blok stále nemá 10 ověření). V tomto případě klient kontaktuje server, který tuto transakci připodepisoval a nechá si poslat tuto transakci přímo od něho. Následně se tato transakce ověří stejným způsobem jako jakákoliv jiná TX3. Pokud je ověření v pořádku, není hlášeno napadení kontaktu, ale pouze varování, které jakmile blok získá 10. ověření zmizí.

VALIDACE VLASTNÍHO KONTAKTU

Při této validaci se postupuje stejně, jako při validaci kontaktu cizího. Navíc se ověřuje, že přístup k poslední TX3 transakci má daný kontakt.

a. 2. výstup u poslední TX3 transakce je zabezpečen MULTISIG 2 ze 2 skriptem, z něhož jedna adresa patří danému kontaktu
b. nebo 2. výstup u poslední TX3 transakce je zabezpečen FRIENDS skriptem, který daný kontakt vytvořil a podepsal

6.6 REDEEM SKRIPT FRIENDS

Privátní klíče používané pro vytváření TX3 transakcí jsou uložené pouze na klientech. Vždy při registraci dalšího zařízení dochází k jejich přenosu spolu s DH klíčem. Pokud by došlo ke ztrátě tohoto klíče, není možné vytvořit novou TX3 transakci a není tedy možné například změnit šifrovací DH-klíč. Pro tento případ si uživatel může nastavit (jako 2. výstup v TX3 transakci) P2SH script, jehož redeem script je typu FRIENDS.

6.6.1 Vlastnosti

Skript umožňuje definovat adresy přátel, kteří mají pravomoc připojit novou TX3 transakci i bez klíče daného uživatele. Při definici lze nastavit množinu přátel a kolik z nich je potřeba pro připojení nové TX3. Vždy je potřeba i součinnost serveru, který nově vzniklou TX3transakci musí také podepsat.

Skript je vytvořen podle následující předlohy **Ks && (Kc || (X-of-Y Kf1, Kf2, ...))**, kde

- **Ks** - klíč serveru
- **Kc** - klíč kontaktu
- **Kfx** - jednotlivé klíče přátel
- **X** - počet přátel potřebných pro připojení
- **Y** - počet všech přátel

Kf1,..., Kfn musejí být lexikograficky seřazené (podle bip 67)

Příklad skriptu:

PUSHDATA(33)[Ks] CHECKSIGVERIFY DEPTH 1 EQUAL IF PUSHDATA(33)[Kc] CHECKSIG ELSE [X] PUSHDATA(33) [Kf1] PUSHDATA(33)[Kf2] PUSHDATA(33)[Kf3] [Y] CHECK-MULTISIG ENDIF