17. Bezpečnost a šifrování

Obsah

- Symetrické a asymetrické šifrování
- Veřejný a soukromý klíč
- Elektronický podpis
- Certifikát
- SSL/TLS
- SSH
- Otisky (fingerprint)
- Man in the middle útok

Symetrické a asymetrické šifrování

Symetrické šifrování

Symetrické šifrování používá **stejný klíč** pro šifrování i dešifrování dat.

Princip: 1. Odesílatel zašifruje data pomocí klíče 2. Zašifrovaná data jsou přenesena příjemci 3. Příjemce dešifruje data pomocí stejného klíče

Výhody: - Rychlost (10-1000× rychlejší než asymetrické) - Nízká výpočetní náročnost - Vhodné pro šifrování velkých objemů dat

Nevýhody: - Problém s bezpečnou distribucí klíče (jak bezpečně předat klíč?) - Každá dvojice komunikujících stran potřebuje unikátní klíč - Pro n účastníků je potřeba n*(n-1)/2 klíčů

Příklady symetrických šifer: - **AES** (Advanced Encryption Standard) - standard v USA, velikosti klíčů 128, 192 nebo 256 bitů - **DES** (Data Encryption Standard) - starší, dnes nedostatečně bezpečný (56 bitů) - **3DES** (Triple DES) - aplikuje DES třikrát, bezpečnější než DES, ale pomalejší než AES - **Blowfish** - rychlý, volně dostupný, velikost klíče až 448 bitů - **ChaCha20** - moderní proudová šifra, alternativa k AES

Asymetrické šifrování

Asymetrické šifrování používá **pár klíčů** - veřejný klíč pro šifrování a soukromý klíč pro dešifrování.

Princip: 1. Každý účastník má pár klíčů: veřejný a soukromý 2. Veřejný klíč může být volně sdílen (proto "veřejný") 3. Soukromý klíč musí zůstat tajný (jen u majitele) 4. Data zašifrovaná veřejným klíčem lze dešifrovat pouze odpovídajícím soukromým klíčem

Výhody: - Není potřeba sdílet tajný klíč přes nezabezpečený kanál - Lepší škálovatelnost (pro n účastníků stačí n párů klíčů) - Umožňuje digitální podpisy

Nevýhody: - Výrazně pomalejší než symetrické šifrování - Vyšší výpočetní náročnost - Delší klíče pro stejnou úroveň bezpečnosti

Příklady asymetrických šifer: - **RSA** (Rivest–Shamir–Adleman) - nejrozšířenější, založen na faktorizaci velkých čísel - **ECC** (Elliptic Curve Cryptography) - založen na eliptických křivkách, kratší klíče - **DSA** (Digital Signature Algorithm) - standard pro digitální podpisy - **Diffie-Hellman** - protokol pro výměnu klíčů, ne přímo šifra

Hybridní šifrování

V praxi se často používá kombinace obou přístupů:

1. **Asymetrické šifrování** pro bezpečnou výměnu klíče
2. **Symetrické šifrování** pro šifrování samotných dat

Příklad procesu: 1. Klient vygeneruje náhodný symetrický klíč (session key) 2. Klient zašifruje tento klíč pomocí veřejného klíče serveru 3. Klient pošle zašifrovaný klíč serveru 4. Server dešifruje klíč pomocí svého soukromého klíče 5. Další komunikace probíhá pomocí symetrického šifrování s tímto klíčem

Výhody: - Kombinuje bezpečnost asymetrického šifrování s rychlostí symetrického - Řeší problém distribuce klíčů - Používá se v SSL/TLS, HTTPS, SSH a dalších protokolech

Veřejný a soukromý klíč

Pár klíčů je základem asymetrické kryptografie a má několik důležitých vlastností:

Veřejný klíč

- Může být volně sdílen
- Používá se pro šifrování dat, která může dešifrovat pouze majitel odpovídajícího soukromého klíče
- Používá se pro ověření digitálního podpisu vytvořeného soukromým klíčem
- Obvykle je distribuován v rámci certifikátu

Soukromý klíč

- Musí být udržován v tajnosti
- Používá se pro dešifrování dat zašifrovaných odpovídajícím veřejným klíčem
- Používá se pro vytváření digitálních podpisů
- Často je chráněn heslem nebo frází

Ochrana soukromého klíče

Pro zvýšení bezpečnosti se soukromý klíč často chrání **přístupovou frází**: - Klíč je na disku zašifrován pomocí této fráze - Při každém použití klíče je potřeba zadat frázi pro jeho dešifrování - Poskytuje dodatečnou vrstvu ochrany v případě krádeže klíče

Další vlastnosti

- Matematicky je extrémně obtížné odvodit soukromý klíč ze znalosti veřejného klíče
- Délka klíče významně ovlivňuje bezpečnost (běžné délky: RSA 2048-4096 bitů, ECC 256-384 bitů)
- Klíče mají omezenou životnost a měly by být pravidelně obnovovány

Elektronický podpis

Elektronický podpis (digitální podpis) je kryptografický mechanismus, který: - Ověřuje identitu odesílatele (autenticitu) - Zajišťuje integritu dokumentu (detekuje změny) - Poskytuje nepopiratelnost (odesílatel nemůže popřít odeslání)

Princip elektronického podpisu

1. Vytvoření **hashe** (otisku) dokumentu pomocí hašovací funkce (SHA-256, SHA-3, atd.)
2. Zašifrování tohoto hashe pomocí **soukromého klíče** podepisujícího
3. Připojení zašifrovaného hashe (podpisu) k dokumentu

Ověření elektronického podpisu

1. Příjemce vypočítá hash přijatého dokumentu (bez podpisu)
2. Dešifruje přijatý podpis pomocí **veřejného klíče** odesílatele, čímž získá původní hash
3. Porovná vypočítaný hash s dešifrovaným hashem
4. Pokud se shodují, podpis je platný a dokument nebyl změněn

Typy elektronických podpisů

- **Jednoduchý elektronický podpis** - základní forma, např. naskenovaný podpis
- **Zaručený elektronický podpis** - jednoznačně spojen s podepisující osobou, umožňuje její identifikaci

- **Kvalifikovaný elektronický podpis** - zaručený elektronický podpis založený na kvalifikovaném certifikátu, právně ekvivalentní ručnímu podpisu

Časové razítko

Časové razítko je dodatečná bezpečnostní vrstva pro elektronický podpis: - Potvrzuje, že dokument existoval v určitém čase - Vydáváno důvěryhodnou autoritou (Time Stamping Authority - TSA) - Zabraňuje zpětnému antedatování dokumentů - Prodlužuje platnost elektronických podpisů i po vypršení certifikátu

Certifikát

Digitální certifikát je elektronický dokument, který propojuje veřejný klíč s identitou jeho majitele. Funguje jako digitální forma průkazu totožnosti.

Obsah certifikátu

- **Veřejný klíč** majitele
- **Informace o majiteli** (jméno, organizace, e-mail, atd.)
- **Informace o vydavateli** (Certification Authority - CA)
- **Sériové číslo** certifikátu
- **Datum platnosti** (od-do)
- **Digitální podpis vydavatele** (CA)
- **Účel použití** certifikátu (podpis, šifrování, autentizace, atd.)

Hierarchie certifikátů

- **Kořenová certifikační autorita** (Root CA) - nejvyšší autorita, podepisuje certifikáty podřízených CA
- **Podřízená certifikační autorita** (Intermediate CA) - vydává koncové certifikáty
- **Koncový certifikát** (End-entity certificate) - používaný koncovými uživateli a službami

Typy certifikátů podle účelu

- **SSL/TLS certifikáty** - zabezpečení webových stránek a serverů
- **Klientské certifikáty** - pro autentizaci uživatelů
- **Kódové certifikáty** - pro podepisování software
- **E-mailové certifikáty** - pro podepisování a šifrování e-mailů

Ověření certifikátu

1. Kontrola digitálního podpisu CA
2. Ověření, že certifikát nebyl odvolán (CRL nebo OCSP)
3. Kontrola platnosti (datum)
4. Ověření důvěryhodnosti vydavatele (CA)

Formáty certifikátů

- **X.509** - standardní formát pro digitální certifikáty
- **PEM** (.pem, .crt, .cer) - Base64 kódovaný DER s hlavičkami
- **DER** (.der, .cer) - binární formát
- **PKCS#12** (.p12, .pfx) - archiv obsahující certifikát a soukromý klíč, chráněný heslem

SSL/TLS

SSL (Secure Sockets Layer) a jeho nástupce **TLS** (Transport Layer Security) jsou kryptografické protokoly, které zajišťují bezpečnou komunikaci v počítačových sítích.

Základní vlastnosti

- Zajišťuje **důvěrnost** (šifrování dat)
- Zajišťuje **integritu** (detekce změn)
- Zajišťuje **autentizaci** (ověření identity serveru, případně i klienta)
- Vytváří zabezpečený kanál na úrovni transportní vrstvy

Proces navázání SSL/TLS spojení (handshake)

1. **Client Hello** - klient posílá podporované šifrovací sady (cipher suites) a náhodné číslo
2. **Server Hello** - server vybírá šifrovací sadu a posílá své náhodné číslo
3. **Certificate** - server posílá svůj certifikát
4. **Server Key Exchange** - server posílá parametry pro výměnu klíčů (pokud je potřeba)
5. **Certificate Request** - server může požádat o certifikát klienta (volitelné)
6. **Server Hello Done** - server signalizuje konec svých zpráv
7. **Client Certificate** - klient posílá svůj certifikát (pokud byl požadován)
8. **Client Key Exchange** - klient generuje pre-master secret a posílá ho serveru
9. **Certificate Verify** - klient potvrzuje vlastnictví privátního klíče k certifikátu
10. **Change Cipher Spec** - obě strany signalizují přechod na dohodnutou šifru
11. **Finished** - obě strany verifikují úspěšné navázání spojení

Verze SSL/TLS

- **SSL 1.0** - nikdy nebyl veřejně vydán
- **SSL 2.0** (1995) - obsahoval závažné zranitelnosti
- **SSL 3.0** (1996) - vylepšená verze, ale dnes považována za nebezpečnou
- **TLS 1.0** (1999) - nástupce SSL 3.0, s drobnými vylepšeními
- **TLS 1.1** (2006) - lepší ochrana proti útokům
- **TLS 1.2** (2008) - podpora moderních kryptografických algoritmů
- **TLS 1.3** (2018) - zjednodušený handshake, odstraněny zastaralé šifry, rychlejší

Použití SSL/TLS

- **HTTPS** - zabezpečená verze HTTP
- **SMTPS, POP3S, IMAPS** - zabezpečená e-mailová komunikace
- **FTPS** - zabezpečený přenos souborů
- **Webové služby a API**
- **VPN** - virtuální privátní sítě

Bezpečnostní problémy a útoky

- **POODLE** - útok na SSL 3.0
- **BEAST** - útok na TLS 1.0
- **CRIME, BREACH** - útoky založené na kompresi
- **Heartbleed** - zranitelnost v implementaci OpenSSL
- **FREAK, Logjam** - downgrade útoky na slabé šifry

SSH

SSH (Secure Shell) je zabezpečený síťový protokol pro bezpečnou komunikaci přes nezabezpečenou síť. Je primárně používán pro vzdálený přístup k systémům a vzdálenou správu.

Hlavní funkce SSH

- **Bezpečné vzdálené přihlášení** k serverům
- **Zabezpečený přenos souborů** (pomocí SCP, SFTP)
- **Tunelování** - přesměrování portů a vytváření zabezpečených tunelů
- **Spouštění vzdálených příkazů**
- **Vzdálené předávání X11** (grafických aplikací)

Metody autentizace v SSH

1. **Heslo** - základní autentizace pomocí uživatelského jména a hesla
2. **Veřejný klíč** - bezpečnější metoda používající pár klíčů
   - Uživatel generuje pár klíčů (veřejný a soukromý)
   - Veřejný klíč se umístí na server (do `~/.ssh/authorized_keys`)
   - Při připojení uživatel prokazuje vlastnictví soukromého klíče
3. **Hostitelská autentizace** - pomocí klíčů serveru
4. **Kerberos** - autentizace pomocí tiketů

Postup autentizace SSH pomocí veřejného klíče

1. Klient kontaktuje server a sdělí mu, že se chce autentizovat pomocí klíče
2. Server zkontroluje, zda má veřejný klíč klienta v `authorized_keys`
3. Server generuje náhodnou výzvu (challenge) a zašifruje ji veřejným klíčem klienta
4. Klient dešifruje výzvu pomocí svého soukromého klíče a odpoví serveru
5. Server ověří, že odpověď je správná, a povolí přihlášení

Komponenty SSH

- **SSH klient** - aplikace pro připojení k SSH serveru
- **SSH server** (sshd) - démon běžící na serveru, který přijímá spojení
- **Klíčové páry** - soukromý a veřejný klíč pro autentizaci
- **Konfigurace** - nastavení SSH klienta a serveru

Bezpečnostní funkce SSH

- **Šifrování** celé komunikace
- **Integrita dat** - detekce změn
- **Autentizace** klienta i serveru
- **Komprese dat** - volitelná pro zvýšení výkonu
- **Přeposílání portů** - vytváření zabezpečených tunelů

Typické použití SSH

```
# Základní připojení
ssh uzivatel@server.priklad.cz

# Připojení na nestandardní port
ssh -p 2222 uzivatel@server.priklad.cz

# Spuštění příkazu na vzdáleném serveru
ssh uzivatel@server.priklad.cz 'ls -la'

# Tunelování lokálního portu na vzdálený server
ssh -L 8080:localhost:80 uzivatel@server.priklad.cz

# Kopírování souborů pomocí SCP
scp soubor.txt uzivatel@server.priklad.cz:/cesta/kam/ulozit/
```

Otisky (fingerprint)

Otisk (fingerprint) je zkrácená reprezentace delšího kusu dat, typicky veřejného klíče nebo certifikátu, vytvořená pomocí hašovací funkce.

Účel otisků

- **Jednodušší ověření** identity
- **Odhalení manipulace** s klíčem nebo certifikátem

- **Rychlé porovnání** bez nutnosti kontrolovat celý klíč
- **Detekce Man-in-the-Middle útoků**

Vytvoření a použití otisku

1. Veřejný klíč serveru je zpracován hašovací funkcí (SHA-256, SHA-1, MD5)
2. Výsledný hash je zobrazen jako řetězec hexadecimálních znaků nebo ve formátu Base64
3. Uživatel může porovnat tento otisk s dříve známým otiskem pro ověření identity serveru

Příklad otisku SSH serveru

```
SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8
```

Ověření otisku SSH serveru

Při prvním připojení k SSH serveru se klientovi zobrazí otisk veřejného klíče serveru:

```
The authenticity of host 'server.example.com (192.168.1.1)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)?
```

Uživatel by měl ověřit tento otisk pomocí důvěryhodného kanálu (telefonát s administrátorem, dokumentace, atd.) před potvrzením.

Otisky a certifikáty

- Otisky certifikátů umožňují rychlou identifikaci konkrétního certifikátu
- Používají se v rámci správy PKI (Public Key Infrastructure)
- Pomáhají při odvolání certifikátů (revocation)

Získání otisku certifikátu nebo klíče

```
# Získání otisku SSL certifikátu
openssl x509 -in certifikat.pem -fingerprint -sha256 -noout

# Získání otisku veřejného klíče SSH
ssh-keygen -lf ~/.ssh/id_rsa.pub

# Získání otisku klíče SSH serveru
ssh-keygen -lf /etc/ssh/ssh_host_rsa_key.pub
```

Man in the middle útok

Man in the Middle (MITM) je typ útoku, při kterém útočník tajně zprostředkovává komunikaci mezi dvěma stranami, které se domnívají, že komunikují přímo spolu.

Princip MITM útoku

1. Útočník se umístí mezi klienta a server
2. Zachytává a případně modifikuje komunikaci v obou směrech
3. Obě strany si myslí, že komunikují přímo, ale ve skutečnosti komunikují s útočníkem

```
Normální komunikace:
Klient <---> Server

MITM útok:
Klient <---> Útočník <---> Server
```

Realizace MITM útoku

- **ARP spoofing** - útočník podvrhne ARP záznamy v lokální síti
- **DNS spoofing** - podvržení DNS odpovědí
- **Falešné Wi-Fi přístupové body** - vytvoření falešného AP s podobným názvem
- **SSL stripping** - degradace HTTPS na HTTP
- **BGP hijacking** - útok na směrování v internetu

Příklad MITM útoku na SSH

1. Útočník zabrání přímému spojení mezi klientem a serverem (např. pomocí ARP spoofingu)
2. Útočník vytvoří SSH server, který se vydává za cílový server
3. Klient se připojí k útočníkovu serveru v domnění, že jde o legitimní server
4. Útočník se současně připojí k legitimnímu serveru
5. Útočník přeposílá komunikaci a může ji odposlouchávat nebo modifikovat

Ochrana proti MITM útokům

- **Ověření fingreprints** serverů před prvním připojením
- **Kontrola certifikátů** (správný vydavatel, platnost, shoda domény)
- **Upozornění na změny** klíčů nebo certifikátů (např. SSH varování při změně klíče serveru)
- **Použití HTTPS** s HSTS (HTTP Strict Transport Security)
- **Dvoufaktorová autentizace** (2FA)
- **Silné šifrování** a moderní kryptografické protokoly (TLS 1.3)
- **Veřejné klíče** známé předem (např. TOFU - Trust On First Use)
- **Certificate Pinning** - pevně dané certifikáty v aplikaci

Detekce MITM útoku

- Neočekávaná změna certifikátu nebo SSH klíče
- Neplatný certifikát nebo varování prohlížeče
- Neshoda v otisku (fingerprint) serveru
- Neočekávaně nízká kvalita šifrování (downgrade)
- Analýza síťového provozu pomocí specializovaných nástrojů

Shrnutí

- **Symetrické šifrování** používá jeden klíč pro šifrování i dešifrování; je rychlé, ale má problém s distribucí klíče.
- **Asymetrické šifrování** používá pár klíčů (veřejný a soukromý); je pomalejší, ale řeší problém distribuce klíče.
- **Veřejný klíč** může být sdílen, zatímco **soukromý klíč** musí zůstat utajen.
- **Elektronický podpis** používá soukromý klíč k podpisu a veřejný k ověření; zajišťuje autenticitu, integritu a nepopiratelnost.
- **Certifikát** spojuje veřejný klíč s identitou a je podepsán důvěryhodnou autoritou.
- **SSL/TLS** protokoly zajišťují bezpečné spojení mezi klientem a serverem; jsou základem HTTPS.
- **SSH** poskytuje bezpečný vzdálený přístup k systémům a zabezpečený přenos dat.
- **Otisky (fingerprints)** umožňují jednoduché ověření identity serveru nebo klíče.
- **Man in the middle útok** je hrozbou, proti které se bráníme ověřováním certifikátů a otisků klíčů.