dSik - Authentication

Lukas Peter Jørgensen, 201206057, DA4

16. juni 2014

1 Sikkerhedsmål

CAA

CAA står for:

Confidentiality: Information skal holdes hemmelig for uvedkommende, gælder både under forsendelse, opbevaring og behandling af data

Authenticity: Informationen er autentisk, den er ikke blevet manipuleret af en uautoriseret person.

Availability: Systemer skal være tilgængelig når de skal bruges

Definition af et sikkert system

Det er typisk svært eller umuligt at bevise et system er sikkert. Tit bliver man nødt til at lave mange antagelser om angriberens muligheder for at bevise at et system er sikkert og disse antagelser er typisk forkerte.

Vi bruger udtrykket "sikret system"istedet for "sikkert system"da det blot indikerer at vi har sikret systemet mod nogle bestemte tilfælde. Disse tilfælde defineres ved en *Sikkerhedspolitik* på baggrund af en *Trusselmodel*, og implementerer herefter denne sikkerhedspolitik vha. nogle *Sikkerhedsmekanismer*. På den måde får vi, at et sikret system kan beskrives som:

$$Sikretsystem = Sikkerhedspolitik + Trusselsmodel + Sikkerhedsmekanismer$$

2 Kryptologi

Kryptografi: Videnskaben om at lave koder og ciffere.

Kryptoanalyse: Videnskaben om at bryde koder og ciffere.

Secret-key/Public-key Authenticity

Vi må antage at vore fjender altid kender til vore krypteringsalgoritmer, så et cryptosystem må ikke afhænge af at holde algoritmen hemmelig.

Istedet skal man bruge en nøgle, som skal distribueres mellem parterne, og bruges i kombination med krypteringsalgoritmen. Der er overordnet set to måder at bruge nøgler, Secret-key og public-key systemer.

I et cryptosystem til authenticity har man to algoritmer: S, for signing og V for verification.

Secret-key

Man signer en besked m med nøglen k således:

$$c = S_k(m)$$

Så sender man m, c til modtageren således han har både meddelelsen og signaturen. Dette kaldes en MAC. Modtageren kan verificerer signaturen c vha. V således:

$$V_k(m, S_k(m)) = accept/reject$$

Dette forhindrer angriberen i at finde en besked m', som ikke tidligere er blevet sendt og som samtidig har en gyldig MAC, medmindre angriberen har en nøgle k.

Public-key

Man signerer således:

$$c = S_{sk}(m)$$

$$V_{pk}(m, S_{sk}(m)) = accept/reject$$

Begge systemer er kan brydes ved *exhaustivekeysearch*, dette sørger man dog for er usandsynligt i praksis. Der findes algoritmer til at finde sk ud fra pk derfor er nøglestørrelserne for public-key kryptering langt større end for secret-key. (Typisk minimum 1024 bit. GPG anbefaler 2048 bit).

Hvorfor bruger man ikke altid public-key? Det er for langsomt.

Replay-attacks bør forhindres ved at bruge et sekvensnummer eller timestamp.

3 MAC

CBC-MAC

Vi blokkene $M_0 ... M_t$, og der er $t+1$ bloks, den ekstra blok er IV'en som ligger ved M_0, og er sat til at være en blok med rene 0'er.

$$C_i = E_k(M_i \bigoplus C_{i-1})$$

$$M_i = D_k(C_i) \bigoplus C_{i-1}$$

MAC'en er så blot den sidste blok.

HMAC

Man bruger en hashing funktion H som f.eks. MD5 eller SHA, og bruger denne til at lave et hash af meddelelsen og nøglen.

$$HMAC_k(m) = H[(k \bigoplus opad)H((k \bigoplus ipad)m)]$$

Hvor nøglen k er plevet padded til at passe til H's bitstørrelse, og hvor *opad* (outer padding) og *ipad* (inner padding) er to 1-blocks hex padding konstanter med værdierne:

$$opad = 0x5c5c5c5c \ldots 5cipad = ox36363636 \ldots 36$$

Sikkerheden afhænger af hashingfunktionen.

4 Hashing

En hash funktion skal:

- Kunne tage en meddelelse af enhver længde som input.
- Producere et output af en fastlagt længde.
- Være hurtig
- Være et svært beregningsmæssigt problem at finde en kollision.

Sværhedsgraden af at finde kollisioner afhænger af hashing teknologien, men det kan vises at en funktion H med k bits output vil der være $2^{k/2}$ gentagelser af H hvis man bruger tilfældige beskeder til at forsøge at finden en kollision. Dette betyder at man bør have et output på min. 160bit i dag.

Man kan udnytte en hash funktion til at tage en MAC på hash-værdien istedet for beskeden, for at beregne MAC'en hurtigere. Man risikerer dog kollisioner så tog beskeder har samme MAC.

RSA

1. Vælg to store primtal p og q og sæt $n = pq$, således n bliver produktet af 2 primtal.
2. Udregn nu $t = \phi(n) = (p-1)(q-1)$, som kaldes Eulers totient af n.
3. Vælg en positiv integer e some er større end 1 og mindre end t, og som er indbyrdes primisk med t. Dette kan også skrives som at $e \in \mathbb{Z}, 1 < e < t, gcd(e,t) = 1$. En måde at gøre dette er at vælge e til at være et primtal. man vil typisk gerne have e til at være lille da det gør krypteringen nemmere og ingen sikkerhedsmæssig betydning har.
4. Udregn d, således at den opfylder kongruens relationen $ed \equiv 1(\mod t)$. Hvilket vil sige at $\mod t$ for $ed - 1$ skal give 0.

Fordi e kan vælges til at være lille, kan man sørge for at krypteringen går hurtigt, dette kan man ikke gøre for d, derfor vil kryptering typisk gå meget hurtigere end dekryptering.

Nu har vi så 3 tal: n, d og e som giver nøglerne $pk = (n, e)$ og $sk = (n, d)$. Så krypterer man ved:

$$c \equiv m^e (\mod n)$$

$$m \equiv c^d \equiv m^{ed} (\mod n)$$

Euler's totient teorem siger os:

$$m^{ed} \equiv m^{de} \equiv m^{t+1} \equiv m^1 \equiv M (\mod n)$$

Angreb på RSA

Exhaustive key search

Prime factorization

Man kan faktorisere n.

Discrete Logarithms

Side-channel attack

Regnetiden for 0 er mindre den for 1, så man kan aflæse tiden eller strømforbruget for at aflæse resultatet.