1 Information Theory and Cryptography

Definition 2.3: Et kryptosystem har *perfect secrecy* hvis $Pr[x|y] = P[x]$ for $x \in \mathscr{P}$, $y \in \mathscr{C}$. Altså at sandsynligheden for, at plaintexten er x, givet vi observerer ciphertexten y har samme sandsynlighed som, at vi vælger plaintexten x.

Det ses ved brug af Bayes' Theorem at $Pr[x|y] = Pr[x] \equiv Pr[y|x] = Pr[y] \forall y \in \mathscr{C}$. Det ses trivielt at for et givent $x_0 \in \mathscr{P}$ hvis $Pr[x_0] = 0 \Rightarrow Pr[x_0|y] = Pr[x_0]$. Samme resultat gælder for $y_0 \in \mathscr{C}$ hvis $Pr[y_0] = 0 \Rightarrow Pr[y_0|x] = Pr[y_0]$, derfor undlader vi at kigge på disse, da de ikke tages i brug. Altså $Pr[y|x] = Pr[y] > 0 \Rightarrow \exists K \in \mathscr{K} : e_K(x) = y$. Det følger så at $|\mathscr{K}| \geq |\mathscr{C}|$, samt at $|\mathscr{C}| \geq |\mathscr{P}|$, siden alle encoding rule e_K er injektive. Følgende Theorem følger fra Shannon:

Theorem 2.4 Antag $(\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$ er et kryptosystem, hvor $|\mathscr{K}| = |\mathscr{C}| = |\mathscr{P}|$. Kryptosystemet giver perfect secrecy $\iff$ alle keys $K \in \mathscr{K}$ bruges med samme sandsynlighed $\frac{1}{|\mathscr{K}|}$ og $\forall x \in \mathscr{P}$ og $\forall y \in \mathscr{C}$ så $!\exists K$ sådan at $e_K(x) = y$

Det er dette theorem der beviser at One-time Pad, har perfect secrecy.

Cryptosystem 2.1, One-time Pad: Lad $n \geq 1$ være et heltal og lad $\mathscr{P} = \mathscr{C} = \mathscr{K} = (\mathbb{Z}_2)^n$. For $K \in (\mathbb{Z}_2)^n$ definer $e_k(x)$ til at være vektor summen modulo 2 af K og x (eller, equivalent, den exclusive-or af de to bit strenge). S¨, hvis $x = (x_1, ..., x_n)$ og $K = (K_1, ..., K_n)$, så

$$e_K(x) = (x_1 + K_1, ..., x_n + K_n) \mod 2$$

Decryption er identisk, så for $y = (y_1, ..., y_n)$, så

$$d_K(y) = (y_1 + K_1, ..., y_n + K_n) \mod n$$

Men det at have perfect secrecy har visse ulemper, specielt det at faktum at $|\mathscr{K}| \geq |\mathscr{P}|$. Det at vi, som fx i One-time pad, skal have en n bit lang key til en n bit lang besked, og denne key ikke kan genbruges til andre plaintexts, gør systemet upraktisk. Altså syntes det nødvendigt at udvikle kryptosystemer som bruger den samme key til forskellige plaintexts. Dette kan gøre systemerne mere sårbare overfor ciphertext-only attacks, til at hjælpe os med dette analysere dette, bruger vi entropy

Definition 2.4 (Entropy): Antag **X** er en stokastisk varibale som kan antage værdier fra en endelig mængde X. *Entropyen* fra den stokastiske varibale **X** er defineret som:

$$H(\mathbf{X}) = -\sum_{x \in X} Pr[x] \log_2 Pr[x])$$

Med denne defination kan man regne entropyen forskellige dele af et kryptosystem, fx $H(\mathscr{K})$, $H(\mathscr{C})$ og $H(\mathscr{P})$.

Som bi-resultat kan nævnes Huffman encodings som er en encoding som er prefix-free og hvor $H(\mathbf{X}) \leq l(f) \leq H(\mathbf{X}) + 1$, hvor $l(f)$ er længden på encodingen.

Definitionen på Entropy kan udvides til også at gælde for betinget entropy.

DEFINITION 2.6: Antag **X** og **Y** er to staokastiske varibale. For alle *fixed* værdier $y \in \mathbf{Y}$, får vi en betinget sandsynlighedfordeling for **X**

$$H(\mathbf{X}|y) = -\sum_{x} Pr[x|y] \log_2 Pr[x|y]$$

Vi definere *betinget entropy*, skrevet som $H(\mathbf{X}|\mathbf{Y})$, som the vægtede gennemsnit (ift. sandsynligheder $Pr[y]$) af entropyerne $H(\mathbf{X}|y)$ over alle mulige værdier for y. Det bergenes som

$$H(\mathbf{X}|\mathbf{Y}) = -\sum_{y}\sum_{x} Pr[y]Pr[x|y] \log_2 Pr[x|y]$$

Den betingede entropy måler det gennesnitlige information-mængde vdr. **X** som ikke afsløres af **Y**

Der existere fundamentale forhold entropyer og componenter i et cryptosystem. Den betingede entropy $H(\mathbf{K}|\mathbf{C})$ kaldes for *key equivocation* og måler mængden af usikkerhed om en key K der er når man kender ciphertexten

THEOREM 2.10 Lad $(\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$ være et kryptosystem. Så er

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C})$$

Vi kan gøre os visse antagelser når omkring den plaintext vi forsøge at finde med ciphertext attacks, bl.a. at vores plaintext er et *natural language*, som fx. engelsk eller dansk. Dette hjælper en evt. adversary med at sortere i mængden af nøgler, som han kan mistænke for at være brugt i krypteringen. Mængden af nøgler der er mulige, men forkerte, kaldes for *spurios keys*. I forsøget på at gøre krypteringer med nøgler af en hvis længe mere sikker, vil vi forsøge at bevise en grænse' for mængden af forventede spurios keys. Til dette vil vi gerne måle mængden af information pr. bogstav i en sætning i et natural language

DEFINITION 2.7: Antag L er et natural language. Lad x være streng af længde n fra L, så er $x \in \mathbf{P}^n$ Entropyen for L er defineret som

$$H_L = \lim_{n \to \inf} \frac{H(\mathbf{P}^n)}{n}$$

og *redundancy* for L er defineret som

$$R_L = 1 - \frac{H_L}{\log_2 |\mathscr{P}|}$$

Forskellige empiriske exiperimenter sætter $1.0 \leq H_L \leq 1.5$ for engelsk. Dette betyder, at den gennemsnitlige mængde af information i engelsk er noget nær 1 til 1.5 bits pr bogstav. Sætter man $H_L = 1.25$ giver dette $R_L = 0.75$ hvilket betyder, at man kan lave Huffman encoding der kan få engelsk til, at fylde omtrent 25% af sin originale længde.

Given en sandsynlighedsfordeling over $\mathscr{K}$ og $\mathbf{P}^n$ kan vi finde en sandsynlighedsfordeling for $\mathscr{C}^n$. Lad $y \in \mathbf{C}^n$, og definer:

$$K(y) = \{K \in \mathscr{K} : \exists x \in \mathscr{P}^n \text{ sådan at } Pr[x] > 0 \text{ og } e_K(x) = y\}$$

altså er $K(y)$ mængden af keys K, hvor y er en encryption af en 'meaningful' plaintext af længden n, altså x fra et natural language fx. Siden vi har brugt en key K til vores encryption er mængden af spurios keys lig $|K(y)| - 1$. Vi angiver den gennemsnitlige antal spurios keys ved $\bar{s}_n$.

THEOREM 2.11 Antag $(\mathscr{P},\mathscr{C},\mathscr{K},\mathscr{E},\mathscr{D})$ er et kryptosystem hvor $|\mathscr{C}| = |\mathscr{P}|$ og keys er valgt med lige stor sandsynlighed. Lad R_l være *redundancy* for det underliggende sprog. Givet en streng af ciphertext af længe n, hvor n er tilpas stor, vil det forventede antal spurios keys, $\bar{s}_n$, opfylde

$$\bar{s}_n \geq \frac{|\mathscr{K}|}{|\mathscr{P}|^{nR_L}} - 1$$

DEFINITION 2.8: *Unicity distance* for et kryptosystem er defineret som værdien n_0, hvor det forventede antal af spurios keys forventes at blive 0. Dvs n_0 bliver længen på ciphertext, som er krævet for at en adversary kan finde en unik nøgle, given nok beregningstid.
Endnu en ting Shannon har bidraget med er ideen om, kombinere cryptosystemer ved, at tage deres 'produkt'. Vi kigger kun på *endomorphic* kryptosystemer, $\mathscr{C} = \mathscr{P}$. Antag $\mathbf{S}_1 = (\mathscr{P},\mathscr{P},\mathscr{K}_1,\mathscr{E}_1,\mathscr{D}_1)$ og $\mathbf{S}_2 = (\mathscr{P},\mathscr{P},\mathscr{K}_2,\mathscr{E}_2,\mathscr{D}_2)$, så er $\mathbf{S}_1 \times \mathbf{S}_2 = (\mathscr{P},\mathscr{P},\mathscr{K}_1 \times \mathscr{K}_2),\mathscr{E},\mathscr{D}$, hvor e_K er difineret o

$$e_{(K_1,K_2)}(x) = e_{K_2}(e_{K_1}(x))$$

og for decryption

$$d_{(K_1,K_2)}(e_{(K_1,K_2)}(y)) = d_{K_1}(d_{K_2}(y))$$

Entropyen for dette nye keyspace beregenes således: $Pr[(K_1,K_2)] = Pr[K_1] \times Pr[K_2]$. Altså da K_1 og K_2 er valgt uafhængigt af hinanden

Et kryptosystem defineres som et *idempotent cryptosystem* hvis $\mathbf{S}^2 = \mathbf{S}$. Shift, substution, Affine, Hill mfl er idempotente kryptosystemer. Hvis et kryptosystem ikke er idempotent er der måske en potentielt gevinst sikkerheds mæssigt ved at lave flere iterationer af samme system.

2 SYMMETRIC (SECRET-KEY) CRYPTO

Vi definere et krypto system som en tre-tupple (G, E, D):

G, ALGORITME TIL AT GENERE KEYS: algoritmen er probabilistisk, tager intet input og outputter altid en key K. For symetriske kryptosystemer bruges key K til både encryption of decryption. Normalt er der defineret to endelige mængder, $\mathscr{P}$ - plaintext og $\mathscr{C}$ - ciphertext. Vi vælger så uniformt en key K fra en endelige mængde $\mathscr{K}$

E, ALGORITME TIL ENCRYPTION: algoritmen tager som input K og et $x \in \mathscr{P}$ og producerer et output $E_K(x) \in \mathscr{C}$. Da E kan være probabilistisk kan to encryptions med samme x og K give forskellige resultater.

D, ALGORITME TIL DECRYPTION: algoritmen tager som input $K \in \mathscr{K}$ og $y \in \mathscr{C}$ og producere et output $D_K(y) \in \mathscr{P}$

For at have et kryptosystem kræves det at $\forall K \in {}'\mathscr{K}$ som outputtes af G, så $\forall x \in \mathscr{P}$ vil $x = D_K(E_K(x))$.

Intet af det forgående fortæller noget om sikkerheden for et kryptosystem. Derfor formalisere visse typer angreb på systemet for at kunne sige noget om, hvor sikkert systemet er. Til dette definere vil en *Adversary*, som kan tænkes som en probabilistisk algoritme A, og et *Oracle* O, som vil svare for visse spørgsmål fra A. Vi definere fire typer angreb:

CIPHERTEXT ONLY ATTACK: en sandsynlighedsfordeling D, for plaintext, er fast og algoritmen A kan afhænge af D. Hver gang A spørger O, returneres $E_K(x)$, hvor x er valgt ift. D, og K er produceret af G (og er fast i tiden angrebet forløber over) .

KNOWN PLAINTEXT ATTACK: En sanfsynlighedsfordeling D, for plaintext, er fast og algoritmen A kan afhænge af D. Hver gang A spørger O returneres x **og** $E_K(x)$ hvor x er valgt ift. D, og K er produceret af G (og er fast i tiden angrebet forløber over).

I de to følgende former for angreb afhænger A ikke af D da, han frit kan vælge hvad han vil sende til O.

CHOSEN PLAINTEXT ATTACK: A kan spørge O og give et vilkårligt $x \in \mathscr{P}$ som input. O returnerer $E_K(x)$, hvor K er produceret af G (og er fast i tiden angrebet forløber over).

CHOSEN CIPHERTEXT ATTACK: A kan spørge O og give et vilkårligt $y \in \mathscr{C}$ som input. O returnerer $D_K(y)$, hvor K er produceret af G (og er fast i tiden angrebet forløber over)

Når A stopper har han et results, som i beste fald er vores nøgle K. Men vi bør også overveje andre delmål for A, fx beregne dele af en plaintext osv.

Det forrige har haft det element i sig, at E kunne have været probabilistisk, men dette er ikke tilfældet for symmetric crypto. I dette tilfælde arbejde vi med deterministic system, hvor to gentagelser af $E_K(x) = y$ med samme K og x resultere i samme y. Dette begrænser symmetric crypto systemers sikkerhed, men de kan stadig være gode at bruges, som dele af crypto systemer som fx i DES. Vi betragte denne slags system hvor hver key K mapper en streng x til et unikt y som værende *function families*. Men da man i DES højst kan have 2^{56} forskellige mappings, hvor det total antal function fra $\{0,1\}^{64}$ der peger på sig selv er $2^{64 \cdot 2^{64}}$ er det ikke rigtige random functions vi bruger. I stedet håber vi på, at A har begrænset beregningskræft og det derfor for ham, vil fremstå tilfældigt. DES encryption med en tilfældig key *virker* tilfældig, og derfor siger vi at DES encryption function er en *pseudorandom* function.

Hvad karakterisere så et godt deterministisk kryptosystem? Egenskaben til, at given en kendt x at encryption y virker tilfældig valgt på trods af x. Dette koncept definere vi mere præcist ved at betragte en famillly of functions $\{f_K | K \in \{0,1\}^k\}$ hvor hvert f_k er en function $f_K : \{0,1\}^n \to \{0,1\}^m$. I dette spil betragter vi en probabilistisk algoritme A, som er placeret i et, af to følgende scenarier, og skal gætte hvilket han er placeret i (med en bit, 0 eller 1):

THE IDEAL WORLD: A får adgang til O_{ideal}, som initialicerer en random mapping R fra $\{0,1\}^n$ til $\{0,1\}^m$ (uniformt valgt fra all sådanne mappings), og A giver inputtet x, og modtager $R(x)$

THE REAL WORLD: A får adgang til O_{Real} vælger et K til fældigt fra $\{0,1\}^k$, og fastholder dette K i resten af spillet. A giver O_{Real} x som input, som svarer med $f_K(x)$.
Ud fra dette lader vi A tale med enten O_{Real} eller O_{Ideal}, og vi lader $p(A,0)$ være sandsynligheden for at A kommunikerer med det ene af vores oracles, og $p(A,1)$ være sandsynligheden for, at A kommunikerer med det andet oracle. Vi definere distinguishing advantage, hvor *meget* A kan se forskel på de to scenarier som:

$$Adv_A(O_0, O_1) = |p(A,0) - p(A,1)$$

Hvis tæt på 0 eller lig 0, har A svært ved at differentiere imellem at være i de to scenarier og vores cryptering i *real* scenariet syntes at være tilfældig valgt, og hvis stor(tæt på 1), vil A med stor sikkerhed kunne afgører hvilket scenarie han er i.

3 Public-key Crypto based on Factoring

4 Public-key Crypto based on Discrete Log and LWE

5 Symmetric (secret-key) Authentication and Hash Functions

6 Digital Signature Schemes