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 $f_K : \{0,1\}^n \rightarrow \{0,1\}^m$. 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.

DEFINITION 1 (PRF SECURITY): Vi siger $\{f_K | K \in \{0,1\}^k\}$ er (t,q,ϵ)-sikker pseudorandom function familiy *(PRF)*, hvis, for en vilkårlig adversary A som højt bergener i t tid, og laver q kald til oraklerne, det holder at $Adv_A(O_{Real}, O_{Ideal}) \leq \epsilon$
Den samme form for spil findes i det tilfælde at vi har med et probabilistisk system at gøre. I dette tilfælde skal A dog *kun* kunne se forskel på om han får sin egenkrypteret streng tilbage, eller en helt tilfældig valgt streng. Vi ser på et kryptosystem (G,E,D) hvor A skal afgøre om han er i et, af to scenarier:

THE IDEAL WORLD: A får adgang til O_{Ideal} som på input $x \in \mathscr{P}$, svarer med $E_K(r)$, hvor K er produceret af G og fixed for helheden af angrabet, og r er en tilfældig valgt streng med samme længde som x.

THE REAL WORLD: A får adgang til et normalt *chosen message attack:* O_{Real} får input $x \in \mathscr{P}$, og svarer med $E_K(x)$, hvor K er genereret af G og fixed for helheden af angrebet.

DEFINITION 2 (CHOSEN-PLAINTEXT ATTACK(CPA)-SECURITY): Vi siger kryptosystemet (G,E,D) er (t,q,μ,ϵ)-sikkert, hvis, for en vilkårlig adversary A som beregner i højst t tid, med q calls til oraklet, med plaintext af længde μ-bits, så holder det at $Adv_A(O_{Real},O_{Ideal}) \leq \epsilon$.
Grunden til parameter μ er at disse systemer kan håndtere strenge af forskellige længer, så for at se hvor meget information A har fået fra O, må vi også kigge på længden af hans spørgsmål til O
Med disse definitioner kan man vise at, det er muligt at lave CPA-sikre schemes, ved at tage udgangspunkt i en block cipher som er PRF-sikker. Et sådan scheme er CBC encryption.

CBC ENCRYPTION Lad (G',E',D') være et PRK-sikkert deterministisk kryptosystem, hvor $\mathscr{P},\mathscr{C}$ er fixed og $\mathscr{P} = \mathscr{C} = \{0,1\}^n$ for et vilkårligt n. CBC er (G,E,D) konstruere fra ovenstående block cipher. Lad $G' = G$. Lad plaintext i CBC være alle strenge af længe n. Denne restriktion er kun for at simplificere da system vil kunne tage strenge af vilkårlig længe med visse modifikationer. For at kryptere en streng vælger vi en n-bit streng y_0, og splitter input x i n-bit strenge $x_1,...,x_t$. Vi definere, for $i > 0$ at $y_i = E_K(y_i \oplus x_i)$. Outputtet er ciphertexten $y_0, y_1, ..., y_t$. Decryption er lige så.

THEOREM 1 Antag (G',E',D') er en (t',q',ϵ')-sikker PRF. Så er CBC encryption baseret på dette system CPA (t,q,μ,ϵ)-sikkert for et vilkårligt q, og for

$$\epsilon = \epsilon' + \left(\frac{\mu}{n}\right)^2 \cdot \frac{1}{2^n}$$

givet at

$$t \leq t', \qquad \frac{\mu}{n} \leq q'$$

Vi ser at resultatet siger forskellen imellem ϵ og ϵ' er antallet af block vi krypterer divideret med 2^n. Grundet til dette er basicly birthday paradox, med at der med stor nok sandsynlighed vil komme 2 blokke som er ens efter krypteringen, hvor med det gfølger at:

$$E_{K_e}(y_{i-1} \oplus x_i) = y_i = y_j = E_{K_e}(y_{j-1} \oplus x_j) \qquad i,j > 0 \implies y_{i-1} \oplus y_{j-1} = x_i \oplus x_j$$

Altså hvis vi sørger for at kryptere noget mindre en $2^{n/2}$ inden vi skifter nøgle så er cirka samme sikkerhed som kryptosystemet CBC-krypteringen bliver baseret på. Beviset for ovenstående foregår ved at, definere et game med real, ideal og et nyt hybrid scenarie, og ser hvad sandsynligheden for en collision er.

Ovenstående Theorem 1 findes også som Theorem 2, for counter (CTR) mode, hvor man hvor encryption blot er $y_i = E_K(y_0 + i) \oplus x_i$, hvor $y_0 + i$ er taget i mod 2^n

3 PUBLIC-KEY CRYPTO BASED ON FACTORING

Imodsætning til symmetric crypto systems, hvor der bruges den samme key til både encryption og decryption, er der to forskellige key i public-key crypto systems. En public og som gives åbent ud til alle, så de kan cryptere text, og en secret som bruges til decrypte texten, som kun kendes af dem der skal kunne decrypte text. Vi definere et sådanne system med en 3-tupel (G, E, D) som defineres som følger:

G, ALGORITME TIL AT GENERERE KEYS: denne algoritme er probabilistisk, tager et input k og giver altid et nøgle par (pk, sk). Udfra pk kan man udlede hvorlede $\mathscr{P}$ og$\mathscr{C}$ skal udformes. Disse behøver ikke at gælde for alle nøgler.

E, ALGORITME TIL AT ENCRYPTE: algoritmen tager som input pk og $x \in \mathscr{P}$ producere $E_{pk}(x) \in \mathscr{C}$. E kan være probabilistisk, og producere forskellige ciphertexts ud fra samme x og pk, og konstruere derfor ciphertext udfra en sandsynlighedsfordeling, baseret på x og K.

D, ALGORITME TIL AT DECRYPTE: Algoritmen tager som input sk og $y \in \mathscr{C}$ og producere et output $D_{sk}(y)\mathscr{P}$. Kan være probabilisk, men er, i de fleste tilfælde, deterministisk.

Vi vil altid kræve af et public key system, at $\forall x \in \mathscr{P} : D_{sk}(E_{pk}(x)) = x$.

Disse krypto systemer gør brug af en såkaldt trapdoor one way function, som har den egenskab at den er injective og hurtigt at beregne, men vanskelig at inverte, med mindre man har noget information, vores secret key. Disse functioner er samlet i en function family som mapper fra alle par af keys, til alle encryptions i det pågældende kryptosystem. Disse er defineret som følger:

DEFINITION 3. Vi siger at en system (G, E, D) danner en familie af trapdoor one-way functions er følgende gøre sig gældende:

1. Algoritmerne (G, E, D) definere et kryptosystem, som tidligere nævnt, som alle kører i polynomisk tid, ift det givne k. Yderligere er E deterministisk, altså E er injektive fra sin inputsmængde til sin outputsmængde.

2. Lad en vilkårlig probabilistisk polynomieltids algoritme A være givet. Lad G med input k, give (pk, sk). Vælge et vilkårligt $x \in \mathscr{P}$, og lad A kører med input $pk, E_{pk}(x)$. Så er sandsynligheden $p(A, k)$, sandsynligheden for at A outputter x ubetydelig lille.
   - En sandsynlighed $\epsilon(k)$ er ubetydelig, hvis, for et vilkårligt polynomie f, gælder $\forall k : \epsilon \leq \frac{1}{f(k)}$. Sandsynligheden går asymptotisk i k, mod 0.

Et sådanne deterministisk kryptosystem er RSA, vi gør os derfor følgende antagelse:

RSA ASSUMPTION: den basale RSA algoritme, fefinerer en familie af trapdoor one-way-permutations.

CRYPTOSYSTEM 5.1: RSA CRYPTOSYSTEM Lad $n = pq$ hvor p og q er primtal, samt $\mathscr{P} = \mathscr{C} = \mathbb{Z}_n$. Lad $\mathscr{P} = \mathscr{C} = \mathbb{Z}_n$, og definer

$$\mathscr{K} = \{(n, p, q, e, d) : ab \equiv 1 \mod \phi(n)\}$$

For $K = (n, p, q, e, d)$, definer

$$e_K(x) = x^e \mod n$$

og

$$d_K(y) = y^d \mod n$$

$(x, y \in \mathbb{Z}_n)$. Public keyen består af værdierne n og e, og secret keyen består af værdierne n og d.

For at kunne bruge dette, og ligende krypto systemer, kræver det, at vi har nogle (meget) store primtal. Disse genereres lave store "*random primes*" og derefter checke med randomized polynomialtime Monte Carlo algoritmer, ved gentagende test, at det er et rigtigt primtal vi har fundet.

DEFINITION 5.1: En *yes-biased Monte Carlo algoritme* er en randomized algoritme for et afgørligheds problem, hvor, hvis svaret er "ja", er det altid korrekt, men ved svaret "nej" kan svaret være ukorrekt ... vi siger yes-biased Monte Carlo algoritmer har en *error probability* lig ϵ, hvis sandsynligheden for fejl i nej-svar er højst ϵ.

En sådanne algoritme er Miller-Rabin algoritmen (The strong pseudo-prime test). Denne algoritmer har kompleksitet $O((logn)^3)$ samt et $\epsilon = \frac{1}{4}$.

write $n-1 = 2^k m$, hvor m er ulige;
vælge et tilfældigt heltal a, $1 \leq a \leq n-1$;
$b \leftarrow a^m \mod n$;
if $b \equiv 1 \mod n$ **then**
  return ("n is prime")
end
for $i \leftarrow 0$ *to* $k-1$ **do**
  if $b \equiv -1 \mod n$ **then**
    return "n is prime"
  else
    $b \leftarrow b^2 \mod n$
  end
end
return ("n if composite")

Et public key system som RSA lider den last, at da man giver alle en public key, kan de selv kryptere data, og begynde en exhaustive search efter den unikke secret key. Der kræves det, hvis systemet skal være sikkert at nøglerne har en hvis størrelse. Stinson foreslår 512kb for e, og 1024kb for n.

Noget andet er, at hvis en skulle få fat i a, er det ikke nok kun at finde et nyt e, men at skifte hele system ud, med et nyt n, man vil kunne factorisere n ud fra e, ved hjælp af **Algorithm 5.10**. RSA factor algoritmen er en Las Vegas algoritme, med en worst-case sandsynlighed for succes på $1-\epsilon$. Altså forventes det, at man skal køre algoritmen $\frac{1}{1-\epsilon}$ gange, for at få det rigtige resultat.

Problemet med determistisk encryption, som er det RSA benytter sig af, er hvis en adversary ved jeg sender en af to beskeder af sted, så kan han kryptere begge systemer på forhånd, og sammenligne med det man sender. Derfor er det nødvendigt, for større sikkerhed, at overveje probabilistiske kryptosystemer. Vi siger et probabilistisk kryptosystemer sematisk sikkert hvis en adversary, ikke kan skelne imellem disse to scenarier:

THE IDEAL WORLD: A og O_{Ideal} får begge input k. O_{ideal} kører $G(k)$ og får (pk, sk), og giver pk til A. A laver $x \in \mathscr{P}$ og giver det til O_{Ideal}, som returnere $E_{pk}(r)$, hvor $r \in \mathscr{P}$ er tilfældigt valgt, med samme længde som x. A ouputter bit b.

THE REAL WORLD: A og O_{Ideal} får begget input k. O_{Real} kører $G(k)$ og modtager (pk, sk) og giver pk til A. A laver $x \in \mathscr{P}$ og giver det til O_{Real}, som returnere $E_{pk}(x)$. A outputter bit b.

DEFINITION 4. Vi siger (G, E, D) er CPA(sematisk)-sikkert, hvis for alle probabiliske polynomiel tids adversaries A, det holder at $Adv_A(O_{real}, O_{Ideal})$ er ubetydeligt ift k.

Definition 4 viser tydeligt, at intet deterministisk public key kryptosystem er CPA-sikkert, da A blot kan kryptere x selv, og sammenligne det outputtet fra O. Dog kan de bruges til, at konstruere nye systemer der er CPA-sikre. Dette ses med

THEOREM 4. Hvis en familie af one-way trapdoor perfmutationes findes, så findes et semantisk sikker probabilistisk public key system.
Et ligende sikkerhedskritere man kan opstille er for chosen ciphertexts:

THE IDEAL WORLD: A og O_{Ideal} får input k. O_{Ideal} kører $G(k)$ og får (pk, sk), tilbage, og giver pk til A. A kan nu sende input streng y til O_{Ideal}, og vil modtage $D_{sk}(y)$. A kan gentage dette sålænge ønsket. Derefter konstruere A $x \in \mathscr{P}$ og sender til O_{Ideal}, som svarer med $y_0 = E_{pk}(r)$, hvor r er en vilkårlig streng, med samme længde som x. A må igen spørge O_{Ideal} med et nyt $y \neq y_0$, hvor O_{Ideal} svarer med $D_{sk}(y)$. Ovenstående gentages sålænge det ønskes af A. Til sidst outputter A med bit b.

THE REAL WORLD: A og O_{Real} får input k. O_{Real} kører $G(k)$ og får (pk, sk), tilbage, og giver pk til A. A kan nu sende input streng y til O_{Real}, og vil modtage $D_{sk}(y)$. A kan gentage dette sålænge ønsket. Derefter konstruere A $x \in \mathscr{P}$ og sender til O_{Real}, som svarer med $y_0 = E_{pk}(x)$. A må igen spørge O_{Real} med et nyt $y \neq y_0$, hvor O_{Real} svarer med $D_{sk}(y)$. Ovenstående gentages sålænge det ønskes af A. Til sidst outputter A med bit b.

DEFINITION 5. Vi siger (G, E, D) er chosen ciphertext (CCA)-sikker, hvis for alle probabilistiske polynomial tids adversaries A, det holder at $Adv_A(O_{Real}, O_{Ideal})$ er ubetydeligt ift. k.

THEOREM 5. Hvis der findes en familie af trapdoor one-way permutationer, så findes der et chosen ciphertext sikkert probabilistisk public-key system.

4 PUBLIC-KEY CRYPTO BASED ON DISCRETE LOG AND LWE

Vi ser nærmere på 3 forskellige beregningsmæssige problemer:

THE DISCRETE LOG (DL) PROBLEM Givet en gruppe G, en generator α og et $\beta \in G$, find et heltal a, så $\alpha^a = \beta$

THE DIFFIE-HELLMAN (DH) PROBLEM Givet en gruppe G, en generator α og α^a, α^b hvor a, b er tilfældigt og uafhængigt valg fra $\mathbb{Z}_t$, beregn α^{ab}

THE DECISIONAL DIFFIE-HELLMAN (DDH) PROBLEM Given en gruppe G, en generator α og $\alpha^a, \alpha^b, \alpha^c$, hvor a, b er tilfældigt og udafhængigt valgt fra $\mathbf{Z}_t$, og c er enten $c = ab$ eller uniformt tilfældigt valgt fra $\mathbb{Z}_t$, gæt hvilken case vi er i.

Nogle ting der binder disse problemer sammen, er deres komplesitet. F.eks. er det let, at se, hvis vi kan finde a i α^a for DL, så kan vi løse DH for $b = 0$

LEMMA 1 DH problemet er ikke *svære* end DL problemet

på samme måde er det oplagt at, hvis man kan løse DH, så kan man løse DDH, ved at beregne α^{ab} og sammenligne dette med α^c, altså

LEMMA 2 DDH problemet er ikke *svære* end DH problemet

Dog er det for begge lemmaer, ikke lykkedes at vise, at de to problemer er helt ækvivalent.

Men hvorledes definere vi at, *et problem er svære end et andet problem?*. Til det må vi først formalisere en *group generator - GGen*. $GGen$ er en efficient probabilistisk algoritme some tager et k som input og outputter en gruppe G og et elementer $\alpha \in G$, som er generator for G. En generator kan findes ved hjælp fra lemma 4:

LEMMA 4 $\alpha \in \mathbb{Z}_p^*$ er en generator $\iff \alpha^{(p-1)/q} \neq 1$ for alle primtal q som deler $p-1$

Pointen med dette er, at hvis vi vil arbejde i en gruppe $\mathbb{Z}_p^*$, så er det muligt, at vælge et p så ingen af vores problemer er *svære*. Derfor kan vi se, at problemer har kun det *hardhed* vi behøver/giver dem, ift. de algoritmer der vælger hvilken gruppe vi er i.

DEFINITION 1 (DL) DL problemet er svært (ift $GGen$) hvis, for en hvilkårlig polynomiel tids(i k) algoritme A, at sandsynligheden for at A outputter a er ubetydelig ift k.

DEFINITION 2 (DH) DH problemet er svært (ift $GGen$) hvis, for en hvilkårlig polynomiel tids(i k) algoritme A, at sandsynligheden for at A outputter αab er ubetydelig ift k.

DEFINITION 3 (DDH) DDH problemet er svært (ift $GGen$) hvis, for en hvilkårlig polynomiel tids(i k) algoritme A, at sandsynligheden for at A har fordel $Adv_A(k) = |p_{A,0}(k) - p_{A,1}(k)|$ er ubetydelig ift k.

Grundet til at netop disse problem er interesant er at de danner grundlag for Diffie og Hellmans foreslag, om et kryptosystem, hvor der var behov for udveksling af nøgler inden kommunikation. Denne idé dannede grundlaget for El Gamal kryptosystemet

EL GAMEL KRYPTOSYSTEM: På input parameter k, generere $GGen$ specifikationerne for gruppen G og genratoren α. Vælg et tilfældigt tal $a\mathbb{Z}_t$, så er vores public key $\beta = \alpha^a$, mens vores secret key er a. Plaintext space er G men ciphertext space er $G \times G$

ENCRYPTION for at encrypte $m \in G$, så vælg et tilfældigt $r \in \mathbb{Z}_t$, og ciphertexten er $(\alpha^r, \beta^r m)$

DECRYPTION For at decrypte en ciphertext (c, d), udregne

$$c^{-a} d = \alpha^{-ar} \beta^r m = \alpha^{ar-ar} m = m$$

El Gamal er semantisk (CPA)sikkert, hvilket vi ser med

LEMMA 3 Problemet med at decrypte en El Gamal ciphertet (uden sk) er ækvivalent til at læse DH problemet.
og da DH er lige så svært som DL som er lige så svært som DDH har vi

THEOREM 1 Hvis DDH problemet er svært (ift. $GGen$), så er El Gamal CPA sikkert.

Her er det vigtigt, at det kun er sikkert ift. $GGen$. Vælges en gruppe $\mathbb{Z}^*$, hvor $p = 2q + 1$, hvor p og q er primtal, så har vi at

LEMMA 6 I gruppen $\mathbb{Z}_p^*$ er DDH problemet aldrig svært

Hvilket skyldes vi kan regne parity af ab, om det er lige eller ulige. Derfor kan man opsætte et spil, hvor A altid gætte rigtigt i real-case, og 50/50 i ideal case, på random streng. Dette resultat skyldes.

LEMMA 5 Lad α være en generator for $\mathbb{Z}_p^*$. Sp er $(\alpha^i)^{(p-1)/2} \mod p = 1$, hvis og kun hvis i er lige, og -1 ellers.

Da $(\alpha^i)^{(p-1)/2} \mod p = (\alpha^{(p-1)/2})^i \mod p = (-1)^i$
Men selv RSA med en gruppe der giver CPA sikkerhed, ville kunne brydes, teoretisk set, af en kvante computer. Derfor må vi komme på alternative metoder til, at beskytte vores data. En sådan metode ville være Learning with error

LEARNING WITH ERROR (LWE): Der generes to nøgler. Secret key, som er en tilfældig vektor $\mathbf{s} \in F_q^n$. Public key som er en række $\{c_i\}_{i=1}^m$, hvor $c_i = (\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + e_i)$, hvor $\mathbf{a}_i$ er uniformly random, og e_i er valgt ift fordeling D_e

ENCRYPTION Beskeden er en bit w. Vælg tilfældige bits $b_1, ..., b_m$ og lad ciphertexten være $\sum_{i=1}^m b_i c_i + (0, \left\lceil \frac{q}{2} \right\rceil w)$

DECRYPTION For at decrypte (u, v), bergn $v - \mathbf{s} \cdot \mathbf{u}$, og output 0 hvis værdien er tættere på 0 end $\left\lceil \frac{q}{2} \right\rceil$. Ellers output 1.

Hvis det antages at decision LWE er svært, så kan man erstatte c_i'erne med public keys, som har formen $(\mathbf{a}_i, u_i)$, hvor u_i er tilfældigt valg, og en adversary ville ikke afgøre dem fra hinanden. Så kan man bevise, at hvis m er stort nok, vil encryptionen af w, med disse nye public keys, stort set ikke have informationer om w i sig

LEMMA 7 Hvis $m \geq n + (n+1)\log_2(q)$, så vil følgende to fordelinger ikke kunne kendes fra hinanden, med kunen ubetydelig, i n, fordel(selv for en unbounded adversary):

$$\{(\mathbf{a}_i, u_i)\}_{i=1}^m \quad \sum_{i=1}^m b_i(\mathbf{a}_i, u_i)$$
$$\{(\mathbf{a}_i, u_i)\}_{i=1}^m \mathbf{r}$$

hvor $\mathbf{r} \in F_q^{n+1}$ er valgt uniformt og uafhængigt fra noget andet.

Dog skal der bruges et q på omkring 1000 og n på nogle hundrede, hvilket giver en ciphertext for $n = 500$, på $(n+1)\log(q) = 5000$bits, hvilket er gangske meget ciphertext for én bit!

5 SYMMETRIC (SECRET-KEY) AUTHENTICATION AND HASH FUNCTIONS

Vi kigger på, fx en situation hvor to parter skal sende data igennem en channel, hvor en adversary kan modificere dataen som han ønsker det. Systemerne vi bruger til, gøre denne kommunikation så sikker som muligt, sender data/message m og muligvis en secret key, eller authentication value s.Her kan vi befinde os i to situationer

ADVERSARY KAN KUN ÆNDRE I MESSAGE En typisk løsning på dette, ville være, at have m et usikkert sted, og have authenticator s et sikkert sted, fx chipcard, også checke at data er intakt, med s. Disse løsninger kan bygges fra *cryptographic hash functions*.

ADVERSARY KAN ÆNDRE BÅDE MESSAGE OG AUTHENTICATOR Dette er tilfældet hvor der kommunikeres over en sikker channel. Vi har to tilfælde i dette

- **Sender og receiver deler en secret key** I dette tilfælde, er authenticator s, lavet på både m og en hemmelig nøgle k. m, s er sendt og vi modtager m', s'. Modtageren kan så ved brug af k, checke om det modtaget ser ok ud. Vi håber at $m, s = m', s'$ og adversarien ikke, har fundet et par andet par m', s', som k vil godtage. Et sådanne system kan bygges på *Message Authentication Codes* (MAC's), også kendte som *Keyed Hash Functions*.
- **Sender og receiver deler på forhånd ingen secret keys** Senderen har en secret key sk, og offentlig gøre en pk. Senderen kommunikere en authenticator s lavet på m og sk. Derefter sendes m, s og m', s' modtages. Nu bruger modtageren pk til, at checke om det modtaget ser ok ud. Et sådanne system kan konstrueres ved brug af *Digital Signature Schemes*.

Vi starter med, at uddybe hashfunctioner.

En hashfunction er defineret ved en *generator* $\mathscr{H}$, som tager som input et sikkerheds parameter k, og outputter beskrivelsen af en function $h : \{0,1\}^* \to \{0,1\}^k$. Med beskrivelse mener vi informationerne, der tillader os, at lave den beregning effektivit, altså i polynomisk tid i k og længden af strengen. Til dette bruges ingen sk, så Adversarien vil også kunne kende den function vi bruger. Praktisk ville vi beregne $h(m)$, som vi gemmer sikkert, og gemme derefter vores message m. Når vi skal bruge vores message, som nu er m', vil vi checke at $h(m) = h(m')$. Vi kalder også $h(m)$ for *message digiest* eller *message fingerprint*.

En adversary kunne have flere angrebsmuligheder mod dette system:

- **Preimage attack:** Given et message fingerprint $h(x)$, finder A en message x så som producere fingerprintet.
- **Second preimage attack:** A finder en message m' så $h(m) = h(m')$.
- **Collision attack:** A finder to vilkårlige beskeder m og m', hvorom det gælder at $h(m) = h(m')$, og går brugeren af systemet til at gemme m, og tage fingerprint $h(m)$, hvorefter han erstatter m med m'.

Det ses at hvis vi kan lave et secound preimage attack, og i nogle tilfælde, hvis vi kan lave et preimage attack, kan vi også lave et collision attack. Derfor er ville den bedste sikkerhed være givet, hvis vi kan sikre os imod collision attack.

DEFINITION 1 Overvej spiller hvor vi kører $\mathscr{H}$ på input k, og får h. Vi giver h som input til A, som outter to strenge m og m'. Vi siger, dette lykkedes hvis $m \neq m'$ go $h(m) = h(m')$. Vi siger $\mathscr{H}$ er *collision intractable*, aka. *collision-resistant*, hvis en vilkårlig polynomieltids algoritme A lykkedes med ubetydelig sandsynlighed (som function af k).

Sådanne functioner findes under *intractability antagelsen*. Antag $\mathscr{H}$ er defineret som følger: vælg et primtal $p = 2q+1$, hvor q er et $k-1$bit primtal. Vælg α, β af orden q i $\mathbb{Z}_p^*$. Definer functionen $h : \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{Z}^*$, ved $h(m_1, m_2) = \alpha^{m_1}\beta^{m_2} \mod p$. Hvis man kan finde en kollision for h, betyder dette vi har $(m_1, m_2) \neq (m_1^{'}, m_2^{'})$ så $\alpha^{m_1}\beta^{m_2} = \alpha^{m_1^{'}}\beta^{m_2^{'}}$. Vi har enten at $m_1 \neq m_1^{'}$ eller $m_2 \neq m_2^{'}$. Lad os antage, uden tab af generality, at vi har i første case. Dette betyder at $(m_1^{'} - m_1)^{-1} \mod q$ findes. Dette betyder at $\alpha = \beta^{(m_2 - m_2^{'})(m_1^{'} - m_1)^{-1} \mod q} \mod p$, og vi har fundet the discrete logarithm for α base β.

Denne konstruktion er dog ikke fuldendt, da den ikke kan håndtere vilkårlige længde af input, som vores definition siger. Dette redes dog af

THEOREM 1 Hvis der findes en collision-intractable hash function generator $\mathscr{H}'$ der producere functioner med endeligt input $m > k$ og med $t(k)$ og $\epsilon(k)$-sikkerhed, så findes der collision-interactable generator $\mathscr{H}$ som producere funktioner, der tager arbitrerer længder af input.
Dette resultat følger af

THEOREM 4.8 Antag **compress** : $\{0,1\}^{m+t} \to \{0,1\}^m$ er en collision resistant compression function, hvor $y \geq 1$. Så findes der en collusion resistant hash function

$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \to \{0,1\}^m$$

Hvor antallet af gang **compress** beregnes i beregningen af h er højst

$$\begin{cases} 1 + \left\lceil \frac{n}{t-1} \right\rceil & \text{hvis } t \geq 2 \\ 2n+2 & \text{hvis } t = 1 \end{cases}$$

hvor $IxI = n$
Vi forsætter med at uddybe MACs

Et *secret-key authentication system*, eller message authentication code(MAC) scheme, består af tre probabilistiske algoritmer (G, A, V).

- **G** - outputter key K, som oftes blot er en tilfædig bit streng af en hvis længde

- **A** - tager en input message m og en key K og producere authenticator value $s = A_K(m)$. s kaldes også for MAC
- **V** -tager som input en authenticator værdi s, en besked m og en key K, og outputter $V_K(s,m)$ som enten er *accept* eller *reject*.
  - det må altid gælde at $V_K(A_K(m),m) = accept$

DEFINITION 2 (CMA SECURITY FOR MACS) Vi siger at et MAC scheme er (t,q,ϵ) CMA-sikkert, hvis, for en hvilkårlig adversary kører højst i tid t og laver højst q queries, vinder et spil, hvor adversary kan spørge et oracle om resultater for $A_K(m)$ og $V_M(a,m)$, for vilkårlige messages m, og hvor han til sidst skal give en message m_0, som oracle ikke har authenticated før og får resultatet $V_K(a_0,m_0) = accept$.
Et af de mest kendte MAC scheme er CBC-MAC, som er baseret på konventionelle kryptosystemer: given et system, fx DES, så kan man blot encrypte data på input message i CBC mode, med $IV = 0$, og lade MAC'en være den sidste block af ciphertext. Hvis man laver sig nogle antagelser om hvilket kryptosystem vi basere CBC-MAC på, fx at det skaber en god familie af pseudorandom funktioner, så vil den resulterende MAC også være godt, til en hvis grad.

"THEOREM Antage (G,E,D) er et (t',q',ϵ')-sikkert PRF, så, for en vilkårlig message af hjøst længde n, antaget ingen message er et prefix an en anden message, og $n < 2^{k/3}$, så vil en CBC-MAC baseret på dette system være (t,q,ϵ) CMA-sikkert MAC scheme, hvor

$$t = t' \qquad t \leq q' \qquad \epsilon = \epsilon' + \frac{20nq'^2}{2^k}$$

og hvor t er det totale antal blokke sendt af adversary igennem angrebet.

Hovedkonklusionen på dette theorem er, så længe vi ikke bruger systemer på for stor mængde data, vil det *arve* mange af de gode styrker fra den original kryptering.

6 DIGITAL SIGNATURE SCHEMES

Vi definere et digital signature scheme ved en tretupel (G, S, V)

- **G:** En probabilistisk key generation algoritme G, som tager et sikkerheds parameter k som input, og producere et key pair (pk, sk). Jo større k, jo, forhåbenlig, bedre sikkerhed.
- **S:** En algoritme A får input, en message m og en secret key sk, og producere en signatur $S_{sk}(m)$.
- **V:** V tager input, signatur s, en message m og en public key pk, og ouputter $V_{pk}(A_{sk}(m), m) = accept$ eller $reject$.
  - Pr def. skal det altid gælde at $V_{pk}(A_{sk}(m), m) = accept$.

Det ses, at imodsætning til MACs, hvor begge parter kende sk, er det kun en bruger der har sk i digital signature schemes.

DEFINITION 3 (SIKKERHED FOR SIGNATURE SCHEMES) Vi sgier et signature scheme er *CMA*-sikkert, hvis for en vilkårlig probabilistisk polynomieltids adversary E, kan vinde spillet med en $Adv_E(k)$, der er ubetydelig ift k: G får input k og producere (pk, sk), og E får pk. E kan quere vilkårligt mange messages m til oraclet O, og får $S_{sk}(m)$ tilbage. E vinder spillet hvis han kan producere et par m_0, s_0 så $V_{pk}(m_0, s_0) = accept$, og O ikke har set m_0 før. Sandsynligheden for at E vinder er en function over k, $Adv_E(k)$. Denne sikkerhed, er den stærkested der kan gives ift signature schemes.

Et sådanne system er fx ElGamal Signature Scheme

KRYPTOSYSTEM 7.2 (ELGAMAL SIGNATURE SCHEME): Lad p være et primtal så det discrete log problem i $\mathbb{Z}_p$ er intractable, og lad $\alpha \in \mathbb{Z}_p^*$ være et primitivt elemet. Lad $\mathscr{P} = \mathbb{Z}_p^*, \mathscr{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ og definer

$$\mathscr{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \mod p\}$$

værdierne p, α og β er vores public key, og a er vores private key.

For $\mathscr{K} = (p, \alpha, a, \beta)$ er for et (hemmelig) tilfældigt tal $k \in \mathbb{Z}_{p-1}^*$, definer

$$\mathbf{sig}_K(x, k) = (\gamma, \delta)$$

hvor

$$\gamma = \alpha^k \mod p$$

og

$$\delta = (x - a\gamma)k^{-1} \mod p-1$$

For $x, y \in \mathbb{Z}_p^*$ og $\delta \in \mathbb{Z}_{p-1}$, definer

$$_K(x,(\gamma,\delta)) = sandt \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \mod p$$

Det er muligt i dettei dette system at lave existential forgery ved et key-only attack, hvor der vælge γ, δ og x på samme tid. Derved vides der ikke hvad der underskrives på forhånd, men det vil ikke desto mindre være underskrevet.

Antag $0 \le i \le p-2$ og $0 \le j \le p-2$ er heltal, og antag $\gamma = \alpha^i \beta^j \mod p$. Vi verificerr ved

$$\alpha \equiv \beta^\gamma (\alpha^i \beta^j)^\delta \mod p$$

som er ækvivalent til

$$\alpha^{x-i\delta} \equiv \beta^{\gamma + j\delta} \mod p$$

Hvor den sidste congruence opfylder

$$x - i\delta \equiv 0 \mod (p-1) \qquad \text{og} \qquad \gamma + j\delta \equiv 0 \mod p-1$$

Given i og j, så kan vi let l'se disse to congruencer modulo $p-1$ for δ og x, givet at $\gcd(j, p-1) = 1$, hvormed vi får følgende

$$\begin{aligned} \gamma &= \alpha^i \beta^j \mod p \\ \delta &= -\gamma j^{-1} \mod (p-1) \qquad \text{og} \\ x &= -\gamma i j^{-1} \mod (p-1) \end{aligned}$$

Som danner vores konstruede signerede message x.

Det er også vigtigt at understrege, at hvis k bliver kendt og $\gcd(\gamma, p-1)$, så følger det at

$$a = (a - xk\delta)\gamma^{-1} \mod (p-1)$$

og systemet vil være fuldstændigt brudt.

En anden fejl er at bruge det samme k for to forskellige messages m, da man så vil kunne beregne a. Det følger sålede: antag (γ, δ_1) er signatur for x_1 og (γ, δ_2) er signatur for x_2, så

$$\beta^\gamma \gamma^{\delta_1} \equiv \alpha^{x_1} \mod p \qquad \text{og} \qquad \beta^\gamma \gamma^{\delta_2} \equiv \alpha^{x_2} \mod p$$

så

$$\alpha^{x_1 - x_2} \equiv \gamma^{\delta_1 - \delta_2} \mod p$$

som er ækvivalent med

$$x_1 - x_2 \equiv k(\delta_1 - \delta_2) \mod (p-1)$$

Lad $d = \gcd(\delta_1 - \delta_2, p-1)$. Da $d|(p-1)$ og $d|(\delta_1 - \delta_2)$, følger det at $d|(x_1 - x_2)$. Definer

$$x' = \frac{x_1 - x_2}{d} \qquad \delta' = \frac{\delta_1 - \delta_2}{d} \qquad p' = \frac{p-1}{d}$$

så bliver congruencen

$$x' = xk\delta' \mod p'$$

og da $\gcd(\delta', p') = 1$, beregner vi

$$\epsilon = (\delta')^{-1} \mod p'$$

hvor vi kan bestemme k modulo p' til

$$k = x'\epsilon \mod p'$$

hvilket giver d kandidater for k

$$k = x'\epsilon + ip' \mod p'$$

for et $0 \leq i \leq d-1$. Af disse d kandidater, er det én unik korrekt kandidat, der kan findes ved at udregne

$$\gamma \equiv \alpha^k \mod p$$

En afart af ElGamal signature er Schnoor Signature scheme, Cryptosystem 7.3, som kan laves med kortere keys, ved at integrere en hash function ind i krypteringen. En sådan integration hjælpe også på El Gamal, delen da den lidt midste sin multiplicative egenskab. Desværre har vi ikke andet hjælp ift at bevise sikkerhed med hash funktioner end random oracle proof. Det bedste vi kan håbe på er

THEOREM 3 Hvis hash functions generator $\mathscr{H}$ er kollision interactable og signature scheme Σ er sikkert, så er det kombinerede scheme Σ' sikkert.

Nogle oplagte problemr med signature schemes er, at godt nok kan vi underskrive data, men vi kan ikke se hvornår. Dette åbner dåren for replay attacks. Dette kan undgås ved at ikke sende den samme besked mere end en gang, numerere sin beskeder, timestamps eller modtageren kan sende et tilfældigt tal R, hvor vi så sender en MAC af R med tilbage. Dette R kan selfølgelig ikke genbruges heller.