Onderzoeksidee - Improving AES by using non-binary number fields

Constantin Blach, s4329872

1 Onderwerp

Mijn onderzoek gaat over een mogelijke additie aan het Rijndael ecryptie algoritme, wat ondertussen beter onder Advanced Encryption Standard (AES) bekend is. AES is inmiddels al rond tien jaar het grondprincipe van encrypties, vast gelegd door de U.S. National Institute of Standards and Technology (NIST). Het algoritme is een symmetrische algoritme, en- en decryptie werkt dus met dezelfde sleutel.

2 Probleemstelling

AES gebruikt merdere (afhankelijk van de lengte van de sleutel) encryptie ronden, waar elke ronde drie stappen bevat. Bij deze stappen worden functies gebruikt, die in een specific getallen field werken. Bovendien word aan elke ronde een session key toegevoegt, zodat het achteraf voor een aanvaller zonder key niet te achterhalen is. Voor alle berekeningen wordt een binary Galois-Field(2^n) gebruikt. Dat betekent, dat alle berekeningen voor de encryptie daarna nog *mod* 2^n gerekend worden. Meestal wordt hier van GF(2^8) (bevat getallen van 0 t/m 255) gebruik gemaakt.

Mijn hoofdvraag is dan, of het van voordeel kan zijn, om een ander getallenfield te gebruiken. Vooral zou het handig kunnen zijn, om GF(p) (p is prime) te kiezen.

Daardoor kan het gekozen field misschien rare characteristics opwijzen, waardoor het voor een aanvaller moeilijk kan zijn, om de sleutel uit te vinden.

Ik moet dus zelf eigen functies bedenken, die in een GF(p) berekeningen uitvoeren, waardoor het moeilijk wordt om het te achterhalen. Deze functie moet de bits, de we willen versleutelen, bij elkaar optellen en/of met elkaar vermenigvuldigen. De beste uitkomst zou zijn, als ik een functie vind, waar elke input een andere output heeft. Dat zou al een vooruitgang zijn, omdat we dat nu nog niet hebben.

Een mogelijke deelvraag/fallback kan zijn, om de binary data, die we willen versleutelen, eerst in het goede formaat in GF(p) te krijgen, wat we dan met ons algoritme kunnen versleutelen. Dat zou een probleem kunnen opleveren, omdat we in GF(p) meer getallen hebben, dan maar twee. Natuurlijk moet aan het

eind, na de decryptie weer dezelfde data tevoorschijn komen. Op deze vraag ga ik alleen meer tijd aan besteden, als mijn hoofd onderzoeksvraag te makkelijk of te moeilijk lijkt.

3 Verantwoording

AES zoals het nu is, is al relatief zeker. Een mogelijk aanvaller zou rond 2^{126} berekeningen moeten doen, om de makkelijkste versie van AES te kunnen breken. Dit aantal is misschien nu nog ok, maar in de toekomst, met nog betere rekeningskracht, zou het vast te makkelijk zijn, om aan de gezochte key te komen.

Daarom zou het handig zijn, om het op dit moment meest gebruikte encryptie algoritme uit te breiden en wat zekerder te maken, om al tegen mogelijke aanvallen beveiligd te zijn.

4 Theoretisch kader

Het meest belangrijke concept voor dit werk is natuurlijk hoe AES in elkaar zit, hoe het werkt en wat de mogelijke problemen zijn. Zoals al gezegd, worden in AES merdere encryptie ronden gebruikt, waarin in elke ronde een sleutel (afhankelijk van de master sleutel) gebruikt wordt. De data moet van te voeren in een matrix vorm worden opgeschreven. Elke ronde ziet er als volgt uit:

1. Substitution: Hier wordt een look-up-table gebruikt, om de data te herschrijven.

2. ShiftRows: Elk getal in de rijen van de matrix word geschuivt. De offset (dus hoever ze geschoven worden) hangt hierbij af van de rijen-index.

3. MixColumns: Elke coloumn word vermenigvulidgd met een polynoom. Deze vermenigvuldiging zal voor minder afhankelijkheid tussen in- en output zorgen.

Na elke ronde wordt dan nog een "RoundKey" van de ronden resulaten berekend, en bij het resultaat erbij opgetelt.

In het begin zijn vooral informatie over algoritme belangrijk, zodat ik misschien een mogelijke verbetering of problemen kan vinden. Hier zijn vooral boeken, maar ook papers over geschreven. (1,2)

Omdat AES het meest gebruikte algoritme in het internet is, is er al veel onderzoek na gedaan, zodat het betrouwbaar is en blijft.

Sinds dat AES bestaat, zijn er al wat security problemen gevonden. Bijvoorbeeld kan de manier (key-schedule) van het berekenen van de "RoundKey" gebruikt worden, om de sleutel makkelijker te vinden. Hier waren dan nog "maar" $2^{99.5}$ berekeningen nodig, in plaats van 2^{126}. (4)

De meeste van de concepten achter de aanvallen, zijn maar op AES met 7 of 8 ronden mogelijk. AES gebruikt eigenlijk 10 of 12 ronden, dus veel van de

mogelijke aanvallen zijn maar ideen, die misschien met betere rekeningskracht mogelijk kunnen worden.

Ook zijn er een aantal side-channel attacks gevonden, waar ik op mijn werk verder niet op in ga, omdat het niet met het algoritme zelf te maken heeft. Ik zoek alleen naar de mogelijke verbeteringen in het algoritme en niet in het gebruiken daarvan.

Er zjn ook recente papers, die al met de beveiliging van AES te maken hebben. Er worden al een aantal problemen met AES gevonden, die tot nu toe nog niet gevaarlijk zijn, omdat we nog te weinig rekenkracht hebben om het te breken. (3)

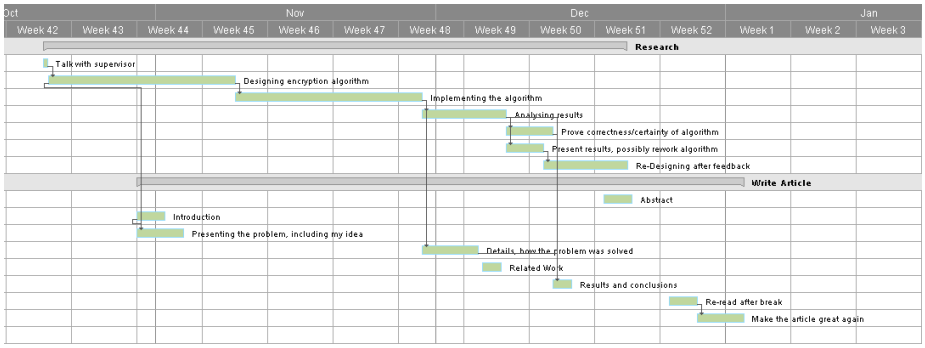
5 Methode

AES gebruikt drie functies per encryptie ronde. Om te kunnen testen, hoe goed mijn bedachte functies werken, moet ik ze dus met de huidige implementatie van AES vergelijken. Dat kan gedaan worden, door de "Diffusion" van de encryptie te bekijken. Voor de zekerheid ervan is het dus belangrijk, hoe groot de relatie tussen in- en output is.

Het beste resultaat, wat we zouden kunnen vinden, is een functie, die 50% van de output verandeerd, als we maar één getal van de input veranderen.

Verder zou ik ook mijn resultaten aan crypto experten kunnen geven, om te kijken, of ze leaking informatie daaruit kunnen halen.

6 Planning



7 Literatuur

1. Paar, C., Pelzl, J.: Understanding Cryptography: A Textbook for Students and Practitioners, 1st edn. Springer Publishing Company, Incorporated (2009)
2. M.Pitchaiah, Philemon Daniel, Praveen: Implementation of Advanced Encryption Standard Algorithm (2012)

3. Joni Moenttinen: The Security of Advanced Encryption Standard (2015)

4. Alex Biryukov and Dmitry Khovratovich: Related-key Cryptanalysis of the Full AES-192 and AES-256 (2009)