PWS Cryptografie - fiks titel nog!

Otto Crawford (6A), Christiaan Tjong Tjin Tai (6B)

8 november 2024

Inhoudsopgave

Hoofdstuk 1

Voorwoord

Hoofdstuk 2

Samenvatting

2.1 Samenvatting in het Nederlands

2.2 Samenvatting in het Engels

Hoofdstuk 3

Inleiding

Hoofdstuk 4

Wat is cryptografie?

Hoofdstuk 5

Wiskunde in de cryptografie

Er zijn een aantal onderdelen in de wiskunde die erg belangrijk zijn in de cryptografie. Met name wiskunde op het gebied van priemgetallen en modulair rekenen is erg belangrijk. Hieronder volgen een aantal definities die wij zullen gebruiken.

5.1 Priemgetallen

Een getal is een *priemgetal* als het getal slechts twee delers heeft: 1 en zichzelf. Deze eigenschap maakt priemgetallen uitstekend bruikbaar in de cryptografie. Als je namelijk twee priemgetallen p en q met elkaar vermenigvuldigt, is er slechts één juiste ontbinding van het product: p*q. Het is erg gemakkelijk voor computers om twee priemgetallen met elkaar te vermenigvuldigen, terwijl het juist veel moeilijker is om een product van twee priemgetallen te ontbinden in zijn factoren. Het is bijvoorbeeld triviaal om 11 en 13 te vermenigvuldigen, maar 143 ontbinden in zijn factoren gaat veel moeilijker. In de cryptografie worden echter niet zulke simpele getallen als 11 en 13 gebruikt, maar priemgetallen van honderden cijfers. Dit zorgt ervoor dat het miljarden jaren duurt om het product hiervan te ontbinden in factoren. Dergelijke operaties met priemgetallen kunnen dus worden gebruikt in de cryptografie, aangezien ze niet gemakkelijk kunnen worden omgekeerd.

5.2 Grootste gemene deler

De *grootste gemene deler* (ggd) van twee getallen, zoals de naam zegt, laat zien wat de grootste deler is die twee getallen gemeen hebben. Deze kan verkregen worden door te kijken naar de priemfactor-ontbinding van beide getallen: de ggd is het product van alle priemfactoren die in beide ontbindingen voorkomen. Als de ggd van twee getallen 1 is, oftewel ze hebben geen priemfactoren gemeen, dan noemen we deze twee getallen *copriem*.

5.3 Modulair rekenen

Modulair rekenen is een vorm van rekenen waarbij elk getal een waarde krijgt tussen 0 en de modulo, n. Deze waarde is gelijk aan de rest die overblijft wanneer het getal door n gedeeld wordt. 11 mod 4 is bijvoorbeeld 3, omdat $11/4 = 2$ rest 3. In de cryptografie wordt modulo rekenen veel gebruikt. Het zorgt er namelijk voor dat bewerkingen van berichten in veel public key-cryptosystemen mogelijk worden gemaakt.

5.4 Inverse

Bij modulair rekenen is de *inverse* van een getal a in modulo n het getal b waarvoor geldt:

$$a \cdot b \, mod \, n = 1 \tag{5.1}$$

Niet alle getallen hebben echter inverses in een bepaalde modulo. Een getal a heeft alleen een inverse in modulo n wanneer geldt dat a en n copriem zijn. Het totaal aantal getallen dat een inverse heeft in een bepaalde modulo, wordt φ genoemd. Voor een modulo n die het product is van twee priemgetallen p en q geldt altijd:

$$\varphi = (p-1)(q-1) \tag{5.2}$$

.

Hoofdstuk 6

Verschillende algoritmes

6.1 Priemgetallen genereren

6.2 Diffie-Hellman???

6.3 RSA

6.4 Digital signatures

6.5 Hashing

Hoofdstuk 7

Een eigen protocol

Dit hoofdstuk betstaat uit twee delen: een omschrijving en de vereisten van het theoretische cryptosysteem [1] en een uitleg van de implementatie hiervan in python.

7.1 Vereisten en implementatie

Als eisen voor het cryptosysteem zijn de volgende punten vastgesteld:

- Alle data wordt opgeslagen op een centrale server: *geen* P2P[2] communicatie.
- Privacy wordt gemaximaliseerd. Zoveel mogelijk informatie moet geheim gehouden worden.
- Zoveel mogelijk algoritmen worden zelf geïmplementeerd.

7.2 Cryptosysteem

Het cryptosysteem dat uiteindelijk is gekozen kan samengevat worden met het volgende diagram: Clients worden een server aangewezen. Elke server heeft een bepaalde hoeveelheid. Hoe meer hoe veiliger, maar hoe meer onnodige data opgeslagen wordt. Praktische tests zijn nodig om de juiste balans hiertussen te vinden.

7.2.1 Stuur-systeem

1. Een client stelt een bericht op om naar de server te sturen.
2. Als eerste wordt de volgende informatie vastgesteld: verzendingstijd, gehashte bericht, bericht, ID van de client.
3. Het bericht wordt versleuteld met RSA via de *public key* van de ontvanger
4. De hash wordt versleuteld met de *private key* van de zender, waardoor het bericht getekend wordt.

7.2.2 Ontvang-systeem

1. Elke paar seconden doet de client een check om te kijken of er nieuwe berichten zijn. Als dit zo is, worden ze opgeslagen.

[1] Moet ik dit definiëren?
[2] P2P: *peer to peer*, betekent dat er een netwerk is van computers, zonder een hierarchie. Denk aan torrents.

2. Een nieuw bericht wordt ontcijferd met de *private key* van de ontvanger. De hash wordt ontcijferd met de public key geassocieerd met de ID die gestuurd is met het bericht.

3. Nu wordt de integriteit van het bericht geverifieerd: door te kijken of de hash van het bericht hetzelfde is als de hash die zojuist ontsleuteld is, kan bepaald worden of het bericht wel of niet juist is (zie figuur 7.1). Op deze manier kan het bericht alleen worden aangenomen als de zender eerlijk is over wie hij is en de ontvanger degene is die de zender bedoelde.

	Bedoelde ontvanger	Onbedoelde ontvanger
Andere zender dan geclaimd	hash ontcijferd met verkeerde *public key*, bericht ontcijferd met verkeerde *private key*. Hash en bericht komt niet overeen en het bericht is wartaal ⇒ wordt weggegooid.	hash ontcijferd met verkeerde *public key* ⇒ zelf berekende hash is niet geleverde hash. Bericht is niet integer en wordt weggegooid
Zender integer	Hash correct ontcijferd, bericht correct ontcijferd, het deterministische hashing algoritme zorgt ervoor dat de berekende hash hetzelfde is als de gestuurde hash. Bericht in orde en wordt opgeslagen.	Hash wordt met de correte *public key* ontsleuteld. Het bericht wordt ontcijferd met de verkeerde private key ⇒ het bericht wordt gelezen als wartaal en de berekende hash komt niet overeen met de geleverde hash.

Tabel 7.1: Caption

Clients Een client stuurt een *packet* naar de server.

7.3 Praktische implementatie

Hoofdstuk 8

Cybercriminaliteit

8.1 Verschillende methoden

8.2 Gevolgen voor ons protocol

Hoofdstuk 9

Quantumcomputers

9.1 Gevaren

9.2 Oplossingen

Hoofdstuk 10

Conclusie

Hoofdstuk 11

Discussie en reflectie

Hoofdstuk 12

fiks bibliografie nog

Hoofdstuk 13

Logboek

13.1 Otto

13.2 Christiaan

Hoofdstuk 14

Verklaring eigen werk