De basis van het verhaal is middelbareschoolwiskunde (ja, één woord, net als *bruinebonensoep*):

$(g^a)^b = g^{ab}$ en $(g^b)^a = g^{ba}$, dus (omdat $ab = ba$): $(g^a)^b = (g^b)^a$.

We stellen $A = g^a$ en $B = g^b$.

Als we er nu voor het gemak even van uitgaan dat niemand ter wereld $a = \log_g A$ of $b = \log_g B$ kan uitrekenen, blijven a en b onbekend als g, A en B publiekelijk worden rondgebazuind en dan kunnen we de volgende procedure gebruiken (rood = geheim, blauw = publiekelijk bekend).

Alice kiest een willekeurig geheim getal a en een niet geheim getal g. Ze stuurt g alsmede $A = g^a$ onversleuteld naar Bob. Iedereen mag het zien. Omdat niemand $a = \log_g A$ kan uitrekenen blijft a toch het geheime getal van Alice, terwijl g en A nota bene in hun helemaal hartstikke blootje zijn verstuurd.	
	Bob ontvangt g en A van Alice en kiest zijn eigen willekeurige geheime getal b. Hij stuurt $B = g^b$ onversleuteld naar Alice. Iedereen mag het zien. Omdat niemand $b = \log_g B$ kan uitrekenen blijft b toch het geheime getal van Bob, terwijl g en B nota bene in hun helemaal hartstikke blootje zijn verstuurd.
Alice ontvangt B van Bob en berekent: $K_A = B^a$ [dat is dus: $(g^b)^a = g^{ba} = g^{ab}$] Daarvoor heeft ze Bobs geheime getal b in 't geheel niet nodig.	Bob berekent: $K_B = A^b$ [dat is dus: $(g^a)^b = g^{ab}$] Daarvoor heeft hij Alice' geheime getal a in 't geheel niet nodig.

Als alles goed is gegaan geldt nu dus $K_A = K_B$ en dat noemen we gewoon K.

Alice en Bob hebben nu een gemeenschappelijk geheim getal K,
terwijl ze malkanders geheime getallen a resp. b niet kennen.

Deze getallen a en b zijn nu nergens meer voor nodig
en die wissen ze dus permanent uit hun geheugen.

Dat is in de cryptografie heel belangrijk: codes voor versleuteling moet je slechts
één keer gebruiken en dan helemaal wegdonderen. Vooral níét hergebruiken!

Deze aldus gevonden K kunnen ze nu als sleutel gebruiken voor hun geheime krommunicatie,
maar slechts gedurende één sessie en dan moet ook K permanent worden gewist.

Voor een eventuele volgende sessie herhalen ze de hele procedure met andere willekeurige
waardes van a en b waaruit dan uiteraard ook een andere waarde van K te voorschijn komt.

De evidente zwakte van dit scenario is natuurlijk dat de logaritmeberekening in werkelijkheid
een fluitje van een cent is en dan kun je er dus naar fluiten want 't is geen fluit waard.
Alice en Bob kunnen op deze manier hun getallen a resp. b helemaal niet
geheim houden en dan loopt $K = g^{ab}$ dus ook pardoes publiekelijk in zijn blootje.

En nu even een raadseltje: ik heb een geheim getal van zeven cijfers en ik verklap de laatste drie. Wat is dan dat geheime getal? Tja, die ander vier cijfers zul je moeten gokken en je mag maar drie keer raden, net als met de PIN-code. Die beschouw jij toch ook als veilig? Domoor!

Die laatste drie cijfers zijn eigenlijk natuurlijk de *rest bij deling* door 1000. Maar waarom zou dat per se zo'n "mooi" getal moeten zijn? Die truc werkt met iedere deler. Laten we die n noemen.

We schrijven: $r = m \bmod n$ (mod staat voor modulo (*klemtoon op 1e lettergreep*)) en dat betekent dat r de rest is die overblijft als je m door n deelt.

Gaat het nu om héél grote getallen, zoals bijvoorbeeld driehonderdduizend miljoen miljard ontelbaar tienduizend biljoen twintig, waarbij m uit veel meer cijfers bestaat dan n, dan treedt bij $r = m \bmod n$ dus een gigantisch cijferverlies op waardoor het onmogelijk is om de waarde van m terug te vinden uit de wijd en zijd bekende getallen r en n. Er rest niets anders dan alle mogelijkheden één voor één (of hoe laat dan ook...) uit te proberen. Chaddamaranstaan. En ondanks de getalgrootte kan de rest bij deling wel degelijk vrij vlot worden bepaald. Er is echter geen begaanbare terugweg van r en n naar m. Er zijn gewoon veel te veel cijfers verdonkeremodulomaand.

En nu hebben slimmeriken het volgende uitgevogeld. Als n gelijk is aan 2, 4, p^k of $2p^k$ (met p een oneven priemgetal) én g is een zogeheten *primitieve wortel* modulo n (dat beperkt de keuze, maar i.i.g. is $g < n$ en ze hebben geen gemeenschappelijke delers; verder moet je 't maar geloven), dan kan voornoemde logaritmetruc $(g^a)^b = (g^b)^a$ ook worden toegepast met gebruikmaking van de moduloberekening! Het wordt dan:

$$(g^a \bmod n)^b \bmod n = (g^b \bmod n)^a \bmod n$$

waarbij we voor n gewoon een (groot) priemgetal p kiezen. In de praktijk is g meestal een niet zo heel erg groot getal dat in weinig tijd kan worden gevonden. Nú moet je om a of b terug te vinden uit A resp. B ineens de zogeheten *discrete logaritme* bepalen en dáárvoor bestaat géén fluitje van een cent, zelfs niet van een miljoen. De enige methode is om álle mogelijkheden uit te proberen. In geval van een héél grote n (denk gerust aan een paar honderd cijfers) is dat zelfs voor een supercomputer onhaalbaar, dus nu zijn het de *spionnen* die ernaar kunnen fluiten.

De sleuteluitwisselingsprocedure wordt nu zoals boven, met dien verstande dat Alice nu dus eerst een willekeurig (heel groot) priemgetal p opzoekt en een bijpassende g bepaalt. Dan kiest ze een (willekeurige) $a < p$. Vervolgens verstuurt ze $A = g^a \bmod p$ naar Bob in plaats van $A = g^a$. Ze moet ook p meesturen, want Bob moet een (willekeurige) $b < p$ kiezen en $B = g^b \bmod p$ uitrekenen in plaats van gewoon $B = g^b$. Daarenboven moeten ze K nu ook via de moduloberekening bepalen, dus $K = B^a \bmod p$ resp. $K = A^b \bmod p$.

Al met al gaan g, p, A en B nu dus helemaal bloot over de lijn, terwijl a, b en K écht geheim blijven.

> Ter illustratie: de snelste computer ter wereld[1] zou sinds de oerknal ongeveer 200 kwintiljard (36 cijfers) zogenoemde *floating point operations* hebben kunnen doen. Een schijntje. Als ie 'n miljard (een 1 met 9 nullen) keer zo snel was, zou die teller vandaag uit 45 cijfers bestaan i.p.v. 36. Nog 'n miljard keer sneller dan, toe maar, alweer 9 nullen erbij. Dat wordt 54 cijfers. Markepperonderd. Hijzerechnoglaaangnioor!
> "Gewone" berekeningen zoals de moduloberekening kunnen echter wél in relatief korte tijd. De leeftijd van het heelal (14 miljard jaar) is $14 \times 10^9 \times 365.25 \times 24 \times 60 \times 60 \approx 0.44$ triljoen seconde, een getal van 18 cijfers. Zo ver kun jij nooit tellen, maar je kunt toch wel zonder rekenmachientje op papier een vermenigvuldiging doen van 2 getallen van 18 cijfers? Een rotklus waar je wellicht een kwartier over doet, oké, maar 't is toch te doen? En voor de moduloberekening doe je een staartdeling tot helemaal onderaan. Da's wat lastiger, maar ach. Uurtje. Maar de *discrete logaritme* van een heel groot getal? Toedeloe!

[1] https://www.datacenterknowledge.com/supercomputers/top500-japan-s-fugaku-still-world-s-fastest-supercomputer

