Diffie-Hellman-nyckelutväxlingsprotokollet

funktion och attacker mot protokollet

Kandidatavhandling i datavetenskap

Sebastian Penttinen

Handledare: Kim Solin

Fakulteten för Naturvetenskaper och Teknik

Åbo Akademi

Åbo 2018

Abstrakt

Diffie-Hellman-nyckelutväxlingsprotokollet används för att säkert kunna skapa delade hemligheter (även kallat nycklar) mellan två olika parter som aldrig tidigare har haft kontakt med varandra. Skapandet av en delad hemlighet sker över osäkra kommunikationskanaler Den delade hemligheten kan sedan användas av parterna för att kryptera sin fortsatta kommunikation. Denna avhandling kommer att fokusera på protokollets grundläggande funktion och de attacktyper som kan användas för att angripa protokollet.

Avhandlingen utförs som en litteraturstudie och börjar med protokollets bakgrund, den bakomliggande teorin och implementationen av protokollet. Sedan går arbetet vidare genom att undersöka olika standarder för protokollet och praktiska användningsområden. Därefter undersöks olika svagheter och attacker mot protokollet.

Nyckelord: Diffie-Hellman, det diskreta logaritmproblemet, Logjam

Innehåll

1 Inledning

Att säkert kunna dela ut nycklar för kryptering kan låta enkelt, men det har varit ett av de svåraste problemen att lösa inom kryptografi. Problemet var tidigare stort; om två parter ville kommunicera säkert med varandra var de tvungna att lita på en tredje part. Denna var nödvändigtvis inte pålitlig och blev då den svagaste länken i kedjan. Problemet med att distribuera nycklarna hävdades vara olösbart, men löstes ändå av matematiker i mitten av 1970-talet. Lösningen anses vara den största revolutionen inom kryptografin på 1900-talet och den största prestationen inom kryptografi sedan man uppfann monoalfabetiska-chiffer (en form av substitutionschiffer) för över 2000 år sedan [26].

Matematikerna som kom med lösningen till nyckeldistributionsproblemet var Whitfield Diffie och Martin Hellman tillsammans med Ralph Merkle [26]. Diffie och Hellman publicerade lösningen i sin forskningsrapport ”New Directions in Cryptography” [7] som utkom 1976 i *IEEE Transactions on Information Theory*. Lösningen på problemet har sedan fått namnet Diffie-Hellman-nyckelutväxlingsprotokollet. I symmetrisk kryptering behöves ett sätt att etablera den delade nyckeln. Symmetrisk kryptering betyder att samma nyckel används för både krypteringen och dekrypteringen. Diffie-Hellman algoritmen fungerar utmärkt för att skapa den delade nyckeln, vilket har lett till att protokollet är i stor användning än idag.

Syftet med avhandlingen är att undersöka hur Diffie-Hellman-nyckelutväxlingsprotokollet fungerar och undersöka de attacktyper som kan användas mot protokollet samt hur dessa kan motverkas. Det finns också Diffie-Hellman-nyckelutväxling över elliptiska kurvor men, de kommer inte att behandlas i denna avhandling.

2 Bakgrund

Under 1960-talet, när datorer blev billigare och kraftfullare, kunde fler och fler företag införskaffa dem för att kryptera sina viktiga digitala kommunikationer, till

exempel banktransaktioner. Som en följd av att allt fler företag införskaffade datorer och kryptering blev vanligare så uppstod ett problem: hur skulle man utbyta krypteringsnycklarna mellan varandra? Detta problem blev känt som nyckeldistributionsproblemet (eng. key distribution problem). Det enda säkra sättet att distribuera nycklar var att fysiskt dela ut dem. Denna process är dock av naturliga skäl tidskrävande. Under 1970-talet försökte bankerna implementera ett system med kurirer istället för personlig utdelning. Kurirerna var betrodda, utvärderade och undersökta anställda. Deras uppgift var att resa runt i världen med låsta portföljer och dela ut nycklar till dem som skulle kommunicera med banken under den kommande veckan. Detta system blev dock snabbt logistiskt ohållbart när bankernas kommunikationsnätverk växte och omkostnaderna blev för höga. Nyckeldistribueringsproblemet är viktigt att lösa, på grund av att till och med det starkaste chiffret kan falla samman om nyckeln till det inte kan distribueras säkert [26].

3 Diskreta logaritmproblemet

För att kunna förstå Diffie-Hellman-nyckelutväxlingsprotokollet måste först det diskreta logaritmproblemet behandlas. Orsaken är att Diffie-Hellman-nyckelutväxlingsprotokollets säkerhet ligger i den matematiska svårigheten att lösa detta problem.

Definitionen på problemet är följande: låt G vara en grupp, skriven multiplikativt, för g ∈ G låt ⟨g⟩ vara den cykliska delgruppen som genereras av g. En grupp är cyklisk om det finns ett g i gruppen som kan uttrycka alla element i G genom g^i (där *i* är ett heltal). När g^i uttrycker alla element i G kallas g för en generator. Alltså g genererar gruppen G [17]. Då kan det diskreta logaritmproblemet anges som:

Givet g ∈ G och a ∈ ⟨g⟩, hitta ett heltal x sådant att

$g^x = a.$

Ett sådant heltal, x, är den diskreta logaritmen av a med basen g. Notationen som används är $x = ind_g a$ (eftersom index är ett annat ord för diskreta logaritmer). Den diskreta logaritmen är endast bestämd för modulo g [16].

Diffie och Hellman var de första att använda sig av det diskreta logaritmproblemet i en kryptografikontext. De föreslog användningen av problemet som en envägsfunktion. Det finns inte någon universellt accepterad definition av en envägsfunktion. En informell definition som kan används är: $f : X \to Y$ för vilket det är enkelt att beräkna $f(x)$ givet $x \in X$, men givet $y \in Y$ är det beräkningsmässigt svårt att räkna ut ett värde x från $f(x) = y$ för de flesta värden på y [16].

Ett enkelt exempel som använder sig av Diffie-Hellmans tänkta sätt att använda diskreta logaritmproblemet som en envägsfunktion är följande.

Exempel 1. Vi har $f(x) = g^x(\ mod\ p)$ och väljer p som ett stort primtal och g som en primitiv rot modulo p. För tydlighetens skull i exemplet väljer vi istället mindre primtal, exempelvis 23 som p och 7 som g, och $7^x\ mod\ 23$, $x\ \in Z$. Svaret kan vara vilken siffra som helst inom (1 , ... , $p-1$). Det är enkelt att beräkna $7^{33}\ mod\ 23 \equiv 22$, men givet $7^x\ mod\ 23 \equiv 22$ är det otroligt svårt att hitta x (diskreta logaritmproblemet). Man skulle behöva prova sig fram och gissa värdet på x. Detta är enkelt för små tal men om det används ett stort primtal som p, som Diffie och Hellman föreslog, så blir problemet extremt svårt att lösa [16].

4 Implementation

Målet med Diffie-Hellman-nyckelutväxlingen är att två användare skall komma överens om en delad hemlighet, oftast är denna en symmetrisk krypteringsnyckel, alltså en teckensträng. Hela utväxlingen kommer att ske genom att endast använda en osäker kommunikationskanal [16].

Att kalla Diffie-Hellman-nyckelutväxlingsprotokollet för nyckelutväxling kan anses lite missvisande, eftersom en nyckel aldrig utväxlas. Utan det är genom publika variabler och deras kombination med privata variabler som den delade hemligheten skapas.

För att beskriva hur nyckelutväxlingen fungerar används vid det här laget branschstandardanvändarna Alice och Bob. Anta att Alice och Bob vill

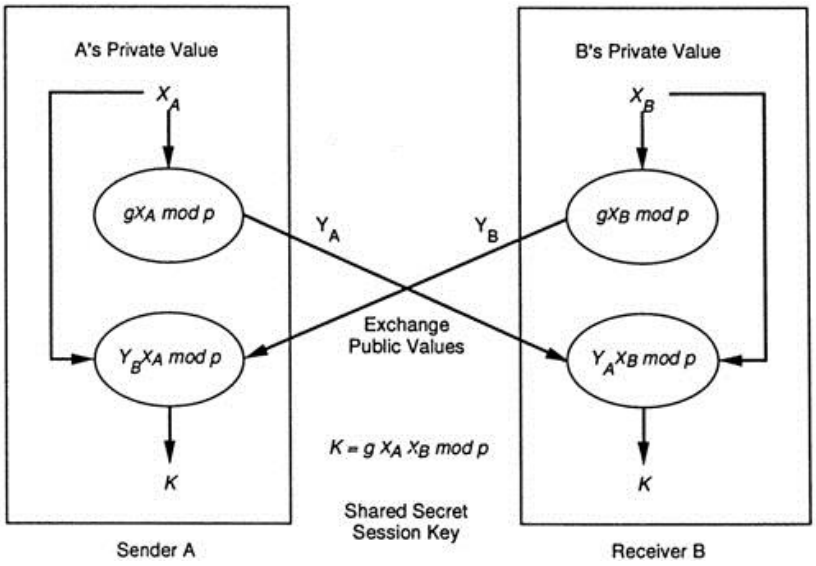
kommunicera säkert med varandra och vill komma överens om en nyckel, för att sedan symmetriskt kunna kryptera sin kommunikation.

Processen att skapa en delad hemlighet börjar med att Alice och Bob kommer överens om två tal p och g [5]. Hur p och q väljs finns listade i RFC 2631 standarden [13]. Viktigaste är att p väljs som ett stort primtal. Den minsta storleken på p som godkänns enligt standarden är 512 bitar (2^{512}). Detta betyder att p är minst ett tal som är 150 siffror långt. Till näst väljer Alice slumpmässigt ett hemligt tal a och Bob väljer också ett slumpmässigt tal b [5]. De slumpmässiga talen skall enligt PKCS 3 standarden [25] väljas enligt $0 < a < p - 1$ och även b skall finnas inom samma intervall.

Alice beräknar sedan sitt publika tal enligt $x = g^a \, mod \, p$. Bob beräknar sitt publika tal på motsvarande sätt enligt $y = g^b mod \, p$. Alice och Bob sänder nu sina publika tal till varandra. Alice sänder x till Bob och Bob sänder y till Alice. Nu beräknar Alice $k_a = y^a \, mod \, p$, alltså Alice beräknar $(g^b \, mod \, p)^a \, mod \, p$, eller kortare $g^{ba} \, mod \, p$. Bob beräknar på motsvarande sätt $k_b = x^b \, mod \, p$, Bob räknar alltså $(g^a \, mod \, p)^b \, mod \, p$ och i kortare form $g^{ab} mod \, p$. Nu har Alice och Bob etablerat den delade hemligheten på grund av att $k_a = \ k_b$[5].

Figur 1, från webbsidan blacksheepnetworks [4], ger oss en förenklad illustration av hela processen. A vill kommunicera med B och skapar ett hemligt, privat tal och ett publikt tal. B gör likadant. Sedan byter A och B de publika värdena med varandra. När A efter detta känner till B:s publika tal och B känner till A:s publika tal kan båda skapa samma delade hemlighet.

Att försöka lista ut den delade hemligheten är opraktiskt. En uttömmande attack skulle behöva göras. Alltså varje tänkbar kombination av $g^{ba} \, mod \, p$ skulle behöva testas. Detta är i teorin möjligt att göra, men i praktiken kommer det att gå ofantligt lång tid innan den rätta kombinationen hittas, förutsatt att primtalen har valts korrekt.



Figur 1: blacksheepnetworks, http://www.blacksheepnetworks.com/security/info/misc/handbook/images/08-04.jpg

5 Standarder

Det finns flera standarder som berör Diffie-Hellman-nyckelutväxlingsprotokollet. Standarderna ger metoder för hur protokollet implementeras, ger testdata och specificerar hur värden skall väljas. Standardernas funktion är att fungera som referenser för hur implementationer av protokollet görs korrekt och säkert.

Till näst beskrivs närmare några standarder i korthet. Standarderna är endast kort beskrivna för det finns hela avhandlingar skrivna om var och en av dessa.

ANSI X9.42 standarden kom ut 1998 och ger instruktioner för hur Diffie-Hellman-nyckelutväxlingsprotokollet skall användas. ANSI X9.42 standarden ger specifika instruktioner för hur protokollet skall användas inom den finansiella sektorn. Standarden instruerar alltså hur den delade hemligheten skall skapas för att få användas för skyddandet av finansiell information. Den delade hemligheten kan också användas för autentisering av finansiell information [2].

RFC 2631 standarden ger instruktioner för en specifik variant av Diffie-Hellman-nyckelutväxlingsprotokollet som bygger på ANSI X9.42 standarden. RFC 2631 är alltså en förkortning av det viktigaste i den större standarden ANSI X9.42. RFC 2631 standarden är därför lättare att följa. RFC 2631 standarden specificerar även

hur p och g skall väljas, som var de publika värden som Alice och Bob har kommit överens om på förhand [13].

PKCS 3 standarden från 1993 beskriver metoder för implementation av Diffie-Hellman-nyckelutväxlingsprotokollet mellan två för varandra tidigare okända parter. Metoderna ger klarhet i hur den delade hemligheten skall skapas. Hemligheten skapas så att den endast är känd mellan två parterna och en tredje tjuvlyssnande part kan inte lista ut nyckeln baserat på kommunikationen mellan de två parterna. Den tänkta användningen för PKCS 3 standarden är att etablera säkra anslutningar exempelvis transportskiktet och nätverksskiktet i OSI modellen [25].

RFC 5114 listar standard uppsättningar med parametrar och testdata som kan användas med Diffie-Hellman. Närmare bestämt åtta stycken. Alla dessa uppsättningar följer de standarder som ISO, ANSI, NIST och IEEE har satt upp [10].

SP 800-56A (version 2) standarden ger rekommendationer för nyckelutväxlings-scheman som bygger på det diskreta logaritmproblemet över ändliga fält eller elliptiska kurvor. Standardens rekommendationer används även som specifikationer för nyckelutväxlingsscheman som anses vara lämpliga att användas av den amerikanska staten [3].

RFC 7836 är en standard som bygger på GOST R 34.10-2012 och GOST R 34.11-2012 standarderna. GOST innehåller de krypteringsalgoritmer som av Ryssland anses vara lämpliga att använda. Diffie-Hellman-algoritmen finns med bland dessa. RFC 7836 standarden är i sig inte en egentlig standard, utan finns för att göra informationen i GOST standarderna tillgänglig för alla [9].

Standarder för kryptering av data i Europa sköts av Enisa (European Union Agency for Network and Information Security). Senaste rekommendation som har publicerats av Enisa är från 2014. Rapporten ger rekommendationer för kryptering och andra säkerhetsåtgärder som behövs för att skydda känsligt material. Rekommendationer för Diffie-Hellman-algoritmen ingår som en del av rapporten [8].

6 Användningsområden

Diffie-Hellman-nyckelutväxlingsprotokollet används som en del av många andra protokoll. Nedan följer några protokoll som använder Diffie-Hellman, eller en variant av Diffie-Hellman, när de skall etablera delade hemligheter för senare kryptering.

6.1 SSH

Secure Shell, eller SSH, är ett protokoll som används för att på distans, över ett nätverk kunna logga in på en dator eller server för att köra kommandon [19].

SSH utvecklades 1995 av finländaren Tatu Ylönen och har sedan dess fått stor spridning. Krypteringen av kommunikationen är SSH:s viktigaste funktion. Liknande protokoll som telnet och ftp (file transfer protocol) krypterar inte någon del av kommunikationen mellan de olika parterna, vilket betyder att till och med lösenord och användarnamn skickas i klartext mellan datorerna. En SSH anslutning börjar med att parametrarna för anslutningen förhandlas fram mellan parterna. När förhandlingen av parametrarna är klar skapas en delad hemlighet med hjälp av Diffie-Hellman-nyckelutväxlingsprotokollet. Efter att hemligheten är skapad använder parterna sig av symmetrisk kryptering [5].

6.2 SSL/TLS

Secure Socket Layer, eller SSL, är ett protokoll för kryptografi som är utvecklat av Netscape, 1995. SSL-protokollet har blivit standarden för säker kommunikation mellan webbanvändare/klienter och webbservrar. Målet är en säker kommunikation för att förhindra tjuvlyssning och försäkra sig om integriteten i kommunikationen, meddelanden kan inte bli ändrade på vägen. Verifikation av att användaren faktiskt är den som den utger sig för att vara ingår även i SSL[5].

TLS, eller Transport Layer Security, är i princip samma sak som SSL. Namnet TLS kommer från att The Internet Engineering Task Force (IETF) accepterade SSL-protokollet år 1999 och döpte om det till TLS [5].

Diffie-Hellman-nyckelutväxlingen används i SSL/TLS:s handskakningsprotokoll. Diffie-Hellman-nyckelutväxlingsprotokollets uppgift är att skapa en delad hemlighet mellan de involverade parterna. Det finns i SSL/TLS flera alternativ för att byta ut nycklar, men Diffie-Hellman anses som det säkraste sättet att utbyta nycklar [5].

Handskakningen i TLS-protokollet börjar med att de involverade parterna kommer överens om vilken typ av krypteringsalgoritm som skall användas. Klienten tar kontakt med servern och meddelar vilka olika krypteringsalgoritmer som denne har möjlighet att använda. Klienten sänder även med ett slumpmässigt tal. Servern väljer sedan en av krypteringsalgoritmerna från listan som klienten har meddelat att den stöder. Därefter sänder servern ett meddelande tillbaka till klienten med ett slumptal, samt vilken krypteringsalgoritm som servern valt att använda. Om Diffie-Hellman har valts för skapandet av den delade hemligheten har servern ansvaret att välja de primtal som kommer att användas. Efter valet av primtal signerar servern digitalt hela paketet, som kommer att skickas tillbaka till klienten. Klienten kontrollerar den digitala signaturen (se kapitel 7.3) för att säkerställa att meddelandet verkligen kommer från servern. Om allt är i sin ordning sänder klinten sin publika del av Diffie-Hellman-nyckelutväxlingen åt servern. För att verkligen kontrollera att allting är korrekt skapar servern och klienten den delade hemligheten och krypterar en MAC (Message authentication code) med den. Servern och klienten utbyter sedan MAC:ar med varandra. Båda parterna kontrollerar sedan MAC:en för att veta om den har blivit ändrad under vägen. Meddelandet med MAC:en fungerar också som en bekräftelse på att handskakningen är färdig och att parterna efter detta kan börja sända meddelanden till varandra [1].

MAC är alltså ett kryptografiskt sätt att upptäcka om ett meddelande har blivit ändrat, antingen av misstag eller avsiktligt [22].

6.3 IKE

IKE, eller Internet Key Exchange, är ett protokoll för IPsec (Internet Protocol Security). IKE används vanligen för säkerheten i virtuella privata nätverk (VPN) [23]. Avhandlingen kommer att endast fokusera enbart på Diffie-Hellman-

nyckelutväxlingsalgoritmens roll i IKE, men IKE stöder även andra protokoll för nyckelutväxling.

Varje kommunikation som använder sig av IKE börjar med en handskakningsprocess. Som en del av processen byts parametrarna för Diffie-Hellman-nyckelutväxlingsprotokollet ut mellan parterna. IKE har tillämpat Diffie-Hellman-algoritmen att byta ut den delade hemligheten som används för varje sänt meddelande för att åstadkomma perfekt framåtsekretess. Det som avses med perfekt framåtsekretess är att om en tjuvlyssnare vill dekryptera vad som har skickats under en kommunikation så behöver tjuvlyssnaren lista ut en ny nyckel för varje enskilt meddelande. Detta betyder i praktiken att varje enskilt meddelande kan vara krypterat med en ny nyckel, vilket gör att om en nyckel listas ut har tjuvlyssnaren endast tillgång till ett meddelande och inte kommunikationen i sin helhet [11].

Perfekt framåtsekretess är enkelt att åstadkomma med Diffie-Hellman-nyckelutväxlingsprotokollet genom enkelheten i utbytet av nycklar som algoritmen erbjuder. Perfekt framåtsekretess är en av de största fördelarna med att använda sig av Diffie-Hellman i krypteringsprotokoll.

7 Attacker mot Diffie-Hellman

Det finns olika attacker som kan riktas mot Diffie-Hellman-nyckelutväxlingsprotokollet. Attackerna utnyttjar olika sorters svagheter i protokollet och nedan behandlas några av dem.

7.1 Generella attacker

Ett av de bästa sätten att försöka lista ut den delade hemligheten är att direkt attackera det diskreta logaritmproblemet. Det diskreta logaritmproblemet gick ut på att lista ut x i $g^x = a$, svaret fås genom att räkna ut $\log_g a$. Det finns algoritmer för att lösa det diskreta logaritmproblemet. Generella algoritmer för problemet fungerar oberoende av grupperna i problemet. När man väljer gruppen som skall attackeras i en generell algoritm brukar man välja ⟨g⟩ som grupp för att ignorera alla element

som finns i G. Variabeln q används för att ange antalet element i ⟨g⟩ och q antas vara känt [14].

En uttömmandeattack mot det diskreta logaritmproblemet kan göras med tidskomplexiteten O(q). Om RFC 3526 rekommendationen [12] följs skulle q vara maximalt 8192-bitar stort därför att det blir opraktiskt att utföra beräkningarna med större tal. Tidskomplexiteten för en generell uttömmandeattack på en 8192-bitar stor grupp tar länge att lösa, men det finns bättre algoritmer att tillämpa på generella grupper [14].

”*Baby-step/giant-step”* algoritmen av Daniel Shanks är en av dessa algoritmer. Den har en tidskomplexitet på O($\sqrt{q} * polylog(q)$) för att lösa det diskreta logaritmproblemet. Algoritmen fungerar på följande sätt: som input tas g och $y \in$ ⟨g⟩. Elementen som finns i ⟨g⟩ föreställs sedan finnas utplacerade i en cirkel enligt följande [14]:

$$1 = g^0, g^1, g^2, \dots, g^{q-2}, g^{q-1}, g^q = 1$$

Någonstans i denna cirkel kommer y att befinna sig. Som tidigare nämnts skulle en uttömmande attack ha en alltför hög tidskomplexitet. Algoritmen delar istället in cirkeln i intervall med $\left[\frac{q}{t}\right] + 1 = O(\sqrt{q})$ element. Cirkeln blir då [14]:

$$g^0, g^t, g^{2t}, \dots, g^{\left[\frac{q}{t}\right]*t}$$

Detta leder till att mellanrummet mellan de olika intervallen i cirkeln är som mest t. Mellanrummen som fås på detta sätt kallas ”giant steps”. I något av dessa mellanrum kommer $y = g^x$ att befinna sig. Vilket leder till att svaret kommer att befinna sig i en av punkterna som delar cirkeln enligt [14]:

$$y * g^0 = g^x, y * g^1 = g^{x+1}, \dots, y * g^t = g^{x+t},$$

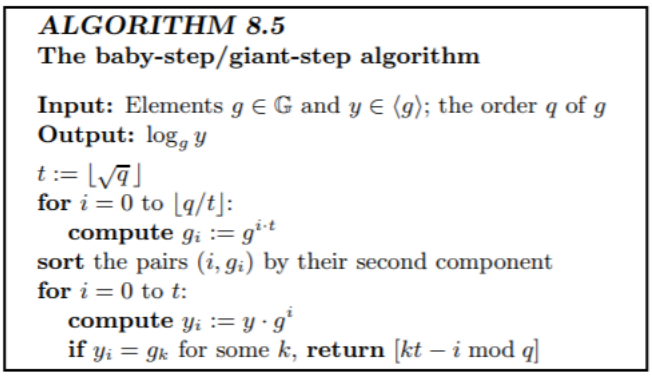
Dessa steg är vad som kallas för ”baby steps”. Om $y * g^i = g^{k*t}$ kan detta lösas genom att hitta $y = g^{kt-i}$ eller $\log_g y = [kt - i \; mod \; q]$ [14].

Katz och Lindell presenterar i sin bok ”*Introduction to Modern Cryptography: Principles and Protocols*” [14] pseudokod för hur algoritmen fungerar (figur 2).

Resultatet av algoritmen är $\log_g y$ och som inputvärden tar algoritmen g och y samt mängden element som finns i q och g. Sedan väljs ett värde *t* som ger värdet av $\sqrt{q}$. Till näst itererar programmet från 0 till q/t där *i* tilldelas värdena som finns mellan dessa intervall. Inne i slingan räknas g^{i*t}och tilldelas g_i. Efter att slingan är klar sorteras paren (i, g_i) enligt g_i. Sedan itererar algoritmen igen från 0 till t och i antar värdena i intervallet. Inne i slingan beräknas $y * g^i$ och tilldelas y_i. Fortfarande inne i slingan kontrollerar algoritmen om y_i och g_k för något *k* antar samma värde inne i slingan. Om detta händer returnerar algoritmen $[kt - i \; mod \; q]$ [14].

Vad algoritmen egentligen gör är att skapa två listor och jämföra värdena mellan dem för att se om det finns likadana värden. Den första listan skapas med endast exponenter till värdet på vänster sida om lika-med-tecknet och på högra sidan om lika-med-tecknet skapas värden enligt högrasidan gånger vänstrasidan upphöjt till en exponent. När algoritmen har hittat ett värde som stämmer tas exponenterna minus varandra och vi har hittat vårt x värde. Genom att titta närmare på följande exempel är det lättare att förstå hur algoritmen fungerar.

Exempel 2 Följande diskreta logaritmproblem $g^x = y \; mod \; 59$ vill lösas med ”baby-step/giant-step” algoritmen. Värdet på x skall hittas. Som input har vi följande värden som befinner sig i den cykliska grupp som bildas av mod 59. $q = 59 - 1 = 58, \; g = 3, \; y = 19$ och som t väljer vi 7. Problemet är alltså $3^x = 19 \; mod \; 59$. Sedan skapas listorna på följande sätt: första listan enligt, $3^0 \; mod \; 59 = 1, \; 3^1 \; mod \; 59 = 3, 3^2 mod \; 59 = 9$ och den andra listan enligt $19 * 3^0 \; mod \; 59 = 19, 19 * 3^1 \; mod \; 59 = 57, 19 * 3^2 \; mod \; 59 = 53$ och så vidare. Efter en stund noteras att $3^{17} \; mod \; 59$ och $19 * 3^0 \; mod \; 59$ ger samma svar nämligen 19. Då kommer svaret att vara $17 - 0$, alltså 17. Testning om detta stämmer, $3^{17} = 19 \; mod \; 59$, ger oss ett korrekt svar.



ALGORITHM 8.5
The baby-step/giant-step algorithm

Input: Elements $g \in \mathbb{G}$ and $y \in \langle g \rangle$; the order q of g
Output: $\log_g y$

$t := \lfloor \sqrt{q} \rfloor$
for $i = 0$ to $\lfloor q/t \rfloor$:
  compute $g_i := g^{i \cdot t}$
sort the pairs (i, g_i) by their second component
for $i = 0$ to t:
  compute $y_i := y \cdot g^i$
  if $y_i = g_k$ for some k, **return** $[kt - i \bmod q]$

Figur 2- Katz Jonathan, Lindell Yehuda Introduction to Modern Cryptography: Principles and Protocols s. 308

7.2 Svaga implementationer

För att det diskreta logaritmproblemet skall vara så svårt att lösa som möjligt måste de värden som Diffie-Hellman-algoritmen använder vara noga utvalda. Primtalet, p, som används bör vara ett säkert primtal. Ett säkert primtal väljs så att $p = 2q + 1$, för ett primtal q. Hur man väljer primtalet q är viktigt för att skapa ett tillräckligt stort primtal p. En vanlig storlek på q är 160-bitar för att skapa ett 1024-bitar stort primtal p. Om man vill ha ett 2048-bitar stort p används ett 224-bitar stort q värde. Primtalet som väljs skall även vara sådant att g genererar en grupp med minst $q \; modulo \; p$ antal element i gruppen [28].

Säkra primtal används alltså för att åstadkomma tillräckligt stora delgrupper. Om ett säkert primtal används kommer storleken på delgrupperna att vara 2, q, eller 2q [1]. Storleken på en delgrupp g fås genom att hitta den minsta exponenten n som gör $g^n = 1$ sant [29].

Beroende på antalet element i den cykliska delgruppen kommer tiden som ”baby-step/giant-step”-algoritmen behöver för att lösa det diskreta logaritmproblemet att variera. Om vi har en liten delgrupp kommer ”baby-step/giant-step”-algoritmen enkelt lösa det diskreta logaritmproblemet. Därför är säkra primtal viktiga.

Enligt rekommendationerna behöver det användas ett dubbelt så stort värde på q än vad som önskas att säkerheten i nyckelutväxlingen skall vara i slutändan. Om målet är att värdet på q skall vara 224-bitar för att få ett 2048-bitars säkerhet kommer det att behöva användas ett 448-bitar stort värde på q för att uppnå en säkerhet på 2048-

bitar. Det behövs ett dubbelt så stort värde på q genom att ”baby-step/giant-step”-algoritmens effektivitet är $\sqrt{q}$ [28].

7.3 Man-i-mitten-attacker

Man-i-mitten-attacker är tjuvlyssning. När information utbyts mellan parter kan tjuvlyssnaren eller ”mannen-i-mitten” snappa upp meddelanden på vägen mellan parterna. Tjuvlyssnaren kan dessutom ändra på de meddelanden som sänds mellan parterna utan att parterna märker detta [20].

Ett exempel på hur en man-i-mitten-attack mot Diffie-Hellman-nyckelutväxlingsprotokollet kan se ut är följande: Tjuvlyssnaren snappar upp de meddelanden där Alice och Bob delar sina publika nycklar med varandra, alltså $g^a \, mod \, p$ och $g^b \, mod \, p$ snappas upp. Tjuvlyssnaren genererar sedan sina egna nycklar att sända tillbaka till Alice och Bob. Detta leder till att Alice tror att den delade hemligheten är $g^{ab´} mod \, p$ och Bob tror att den är $g^{ba´} \, mod \, p$. Nu tror Alice och Bob att de har en delad hemlighet att använda för symmetrisk kryptering. Alice sänder ett krypterat meddelande med $g^{ab´} mod \, p$ som krypteringsnyckel. Tjuvlyssnaren tar emot meddelandet och dekrypterar det genom att Alice egentligen har etablerat sin delade hemlighet med tjuvlyssnaren. Tjuvlyssnaren läser meddelandet och krypterar det sedan igen med $g^{ba´} \, mod \, p$, som är den delade hemligheten som tjuvlyssnaren har med Bob. Efter krypteringen sänder tjuvlyssnaren iväg meddelandet till Bob och varken Alice eller Bob vet om att meddelandet har blivit läst på vägen [17].

Diffie-Hellman-nyckelutväxlingsprotokollet är synnerligen utsatt för man-i-mitten-attacker. En åtgärd som används för att förhindra man-i-mitten-attacker mot Diffie-Hellman-nyckelutväxlingsprotokollet är att använda *digital signering*. Digitalsignering bygger på asymmetrisk kryptering. Asymmetrisk kryptering (exempelvis RSA) betyder att de nycklar som används för kryptering och dekryptering inte är samma nyckel som i symmetrisk kryptering. Nycklarna är istället matematiskt länkade till varandra. En av nycklarna är publik medan den

andra är privat. För att åstadkomma digital signering görs en hashning[1] på den data som man vill signera Efter ett nyckelutbyte med Diffie-Hellman-algoritmen identifieras parterna med hjälp av deras digitala signaturer. Den digitala signeringen gör att tjuvlyssnaren inte kommer att komma över någon information, om inte denne även har den tänkta mottagarens privata nyckel. Parterna märker att tjuvlyssnaren inte har rätt digitala signering [18].

Kritik som riktas mot digitalsignering är att systemet inte är helt vattentätt. Om en privat nyckel stjäls kan den som innehar nyckel utge sig för att vara den ursprungliga innehavaren av den privata nyckeln. De som sedan mottar meddelanden signerade med den privata nyckeln har inget sätt att veta att det inte är den ursprungliga innehavaren som har skickat meddelandet [15].

7.4 Logjam

Logjam attacken är en attack mot TLS-protokollet som utnyttjar flera olika sorters svagheter. Svagheterna upptäcktes av ett forskarteam och en forskningsrapport ”Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice” publicerades i *ACM Conference on Computer and Communication Security*, 2015 [1]. Logjam attacken gör det möjligt för en man-i-mitten-attack att sänka säkerheten i TLS-kommunikationer till 512-bitars exportklassade krypteringsalgoritmer[2]. Attacken är mera en svaghet i TLS-protokollet, men den belyser utmärkt hur svaga implementationer av Diffie-Hellman kan utnyttjas. Dessutom är, enligt forskarteamet, Logjam attacken den första attacken att utnyttja direkta svagheter i diskretalogaritmsystem som är i faktisk användning.

De underliggande orsakerna som gör Logjam attacken möjlig är att ett stort antal servrar på internet fortfarande stöder de osäkra exportklassade

[1] . Hashning är att ändra om en teckensträng av godtycklig längd till en bestämd längd som representerar den ursprungliga teckensträngen [24]. Hashen på indatan krypteras med en privatnyckel. Den krypterade hashen tillsammans med information om vilken hashningsmetod som har använts är en digital signatur [21].

[2] Exportklassade krypteringsalgoritmer härstammar från 1990-talet. Algoritmerna är medvetet gjorda för att gå att knäcka. De är försvagade för att den amerikanska staten krävde att de skulle ha ett sätt att bryta all kryptering som exporterades från USA [27].

krypteringsalgoritmerna eller så använder servrarna svaga implementationer av Diffie-Hellman-nyckelutväxlingsprotokollet. Det är dessutom också vanligt att servrarna använder sig av hårdkodade, standardiserade eller mycket vanliga parametrar i nyckelutväxlingen. Skapandet av primtal med alla speciella egenskaper som behövs för att de skall vara säkra att använda i nyckelutväxlingen kan vara ytterst krävande. Därför finns det standardiserade primtal där detta redan har kontrollerats. Exempelvis Oakley-grupperna är sådana primtal (publicerades, 1998) och de är i aktiv användning i IKE, SSH och flera andra protokoll. De standardiserade och vanliga primtalen leder till att storskaliga attacker blir mera kostnadseffektiva. Om man lyckas bryta en av de vanliga parametrarna kan man läsa av ett stort antal meddelanden [1].

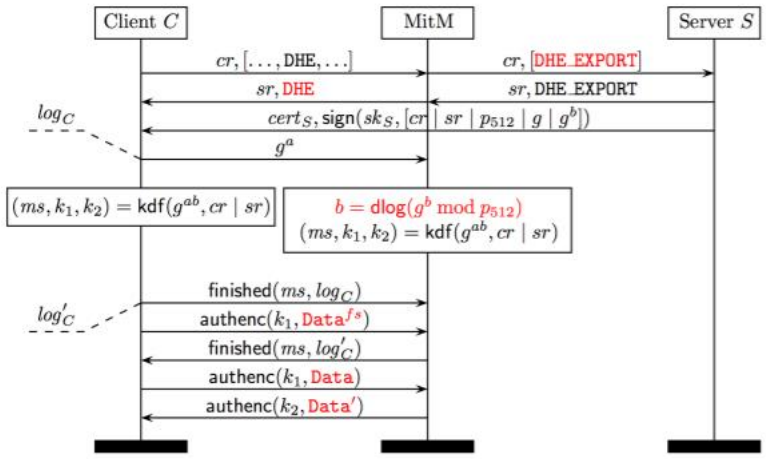
För att lösa det diskreta logaritmproblemet på bästa sätt använde forskarna talfältssållalgoritmer, (eng. ”number field sieve”). De påpekar att det är svårare att lösa diskreta logaritmer gentemot vad det är att faktorisera RSA-nycklar av samma storlek [1].

Talfältssållalgoritmer används för att faktorisera heltal och dessa algoritmer kan även användas för att lösa diskreta logaritmer. Talfältssållalgoritmer är ytterst komplexa och mycket forskning har satts på att utveckla dem. Forskningen runt algoritmerna är avsevärd eftersom upptäckten av en tillräckligt effektiv algoritm leder till att man kan knäcka modern kryptering [6].

Nackdelen som finns med diskreta logaritmer är att om man lyckas lösa en diskret logaritm med ett talfältssåll går det sedan snabbt att lösa vilken diskret logaritm som helst som finns inom samma delgrupp. Detta leder till att kostnaden för att lösa en diskret logaritm går ner markant [1].

Forskarteamet utförde Logjam attacken genom att på förhand använda ett talfältssåll och beräkna diskreta logaritmer för två olika Diffie-Hellman grupper med storleken 512-bitar vardera. Dessa två grupper användes av cirka 92 % av de servrar som var mottagbara för Logjam attacken. Då forskarna på förhand hade löst de diskreta logaritmerna kunde de lösa andra diskreta logaritmer inom dessa grupper på en minut. De olika möjliga logaritmiska lösningarna som på förhand har

räknats fram sparas i en databas och itereras över för att åstadkomma lösningen av de övriga diskreta logaritmerna på en minut[3] [1].



Figur 3: Adrian m.fl. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice s.4

Figur 3 illustrerar hur Logjam attacken utförs i praktiken. En man-i-mitten kan lura TLS-klienter att använda sig av exportklassade krypteringsalgoritmer när de kommunicerar med en server. Tjuvlyssnaren snappar upp det första meddelandet från klienten i TLS-handskakningsprotokollet och ändrar på listan över vilka krypteringsalgoritmer som stöds. Listan ändras så att endast exportklassade krypteringsalgoritmer ingår. Servern har inget att misstänka utan tror endast att det är en äldre klient som vill ha kontakt. Servern svarar med att välja exportklassad Diffie-Hellman och sänder sitt val av algoritm tillbaka till klienten. Mannen-i-mitten ändrar meddelandet på vägen så klienten tror att servern har valt en icke exportklassad 512-bitars Diffie-Hellman-algoritm. Det går inte att se skillnad mellan en exportklassad 512-bitars Diffie-Hellman-algoritm och en icke exportklassad Diffie-Hellman-algoritm på 512-bitar utan det andra meddelande som sänds av server är identiskt i båda fallen. Klienten har därför ingen aning om att meddelandet har ändrats på vägen. Det är också här som svagheten i TLS-protokollet ligger. Om servern skulle ha digitalsignerat sitt val av

[3] Förklaringen till att lösningarna sparas i en databas är förenklad. Förenklingen är gjord för att algoritmen som löser nya diskreta logaritmer inom samma delgrupper är komplex och går utanför avhandlingen.

krypteringsalgoritm så skulle inte mannen-i-mitten kunna ändra på meddelandet utan att klienten skulle märka det [1].

Servern sänder sedan enligt TLS-protokollet iväg ett digitalsignerat meddelande som innehåller parametrarna för Diffie-Hellman-nyckelutväxlingen. Detta meddelande kan inte mannen-i-mitten ändra på utan att bli upptäckt, men behöver inte heller ändra på det för att få attacken att fungera genom att klienten inte kan se skillnad på exportklassad och icke exportklassad Diffie-Hellman. I detta skede av handskakningsprocessen har klienten och servern olika versioner av handskakningen och detta skulle märkas när MAC:en skall skickas. Mannen-i-mitten kan lösa den diskreta logaritmen med hjälp av de beräkningar som har gjorts på förhand. Detta leder till att mannen-i-mitten kan skapa den delade hemligheten och därmed kan verifiera MAC:en med klienten i serverns ställe. Efter detta är mannen-i-mitten helt fri att agera i serverns ställe och kan läsa eller ändra på all data som skickas [1].

Utmaningen för attackeraren som dock finns kvar är att angriparen måste hinna lista ut den delade hemligheten före handskakningsprotokollet har avslutats. Annars kan attackeraren inte förfalska MAC-meddelandet. Forskarteamet lyckades i medeltal lista ut den delade hemligheten på 70 sekunder vilket inte är snabbt nog för att någon av parterna inte skall ana oråd. Det finns dock olika strategier som attackeraren kan använda sig av för att lyckas med sin attack. TLS-klienter, som inte körs via en webbläsare, har vanligtvis inte en bakre gräns på hur länge en handskakning får ta. Kommandotolkar är exempel på sådana klienter. Saknaden av en bakre tidsgräns gör att attackeraren enkelt kan utnyttja sådana TLS-klienter [1].

Webbläsarna däremot tolererar endast att handskakningsprotokollet tar en viss tid. Om attackeraren dock sänder TLS-varningsmeddelanden åt klienten kommer klienten att ignorera varningarna men tidsgränsen för hur länge handskakningen får ta kommer att återställas. Detta leder till att attackeraren kan köpa sig tillräckligt mycket tid för att lyckas hitta den delade hemligheten. Denna strategi menar forskarna att endast kommer att fungera i webbläsaren Firefox, alla andra webbläsare avslutade kommunikationen efter en minut. En attack mot kommunikationen mellan en webbläsare och en server enligt principen att skicka varningsmeddelanden skulle märkas av användaren. Om webbsidan man försöker

komma åt tar över en minut att ladda in stänger de flesta användare sidan. Attackeraren kan dock välja att angripa endast en liten del av webbsidan, exempelvis en annons. Detta leder till att webbsidan kommer att laddas in som vanligt, men den lilla delen av webbsidan som attackerades kommer att laddas in senare [1].

Många servrar som stöder TLS har också svagheten att de inte väljer ett nytt privat tal för varje Diffie-Hellman-nyckelutväxling. Servrarna skapar sitt privata tal en gång (ofta vid uppstart) och använder samma tal upprepade gånger för flera olika kommunikationer. Detta beror på att servrarna är felaktigt konfigurerade. De flesta applikationerna för TLS-servrar har alternativet att skapa nya privata nycklar för varje kommunikation, men administratörerna för servrarna har inte aktiverat funktionen. Svagheten gör, enligt forskarna, det möjligt för en angripare att attackera en kommunikation och lista ut serverns privata nyckel och utnyttja faktumet att den sällan byts ut. I och med att nyckel sällan byts ut kan attackeraren snabbare angripa andra framtida handskakningar. Eftersom serverns privata nyckel är känd behöver attackeraren inte göra några uträckningar utan kan direkt skapa den delade hemligheten [1].

Forskarna undersökte även om det skulle gå att attackera större Diffie-Hellman-grupper med storlekarna 786-bitar och 1024-bitar. Diffie-Hellman grupper med 786-bitar och 1024-bitar anses säkra och är därför i storskalig användning. Forskarna hävdar dock att ett akademiskt forskarteam skulle klara av att lösa diskreta logaritmer med storleken 786-bitar och att storlekar på 1024-bitar möjligtvis är inom räckvidden för en statlig attack. Forskarteamet har undersökt materialet som läcktes av Edward Snowden och drar baserat på det materialet slutsatsen att NSA eventuellt redan klarar av att lösa 1024-bitars Diffie-Hellman-grupper. Om detta är möjligt har det en stor betydelse genom att om redan tio 1024-bitar stora Diffie-Hellman-grupper blir lösta, kan omkring 66 % av all VPN kommunikation med IKE protokollet tjuvlyssnas på och 26 % av all SSH kommunikation, samt 24 % av de mest populära HTTPS webbsidorna [1].

För att åtgärda problemet som Logjam attacken utgör har de mest populära webbläsarna börjat rata för små Diffie-Hellman-grupper. Forskarteamet rekommenderar även att alla TLS-servrar stänger stödet för exportklassade

krypteringsalgoritmer och väljer sina Diffie-Hellman-grupper bättre. Primtal som finns i många standarder skall alltså inte användas utan egna primtal skall skapas. Forskarna uppmanar även att inte medvetet försvaga kryptering, eftersom att hela attacken är möjlig på grund av medvetet försvagade krypteringsalgoritmer [1].

8 Resultat

Det diskreta logaritmproblemet som Diffe och Hellman fick en idé att använda som en envägsfunktion för att skapa en delad hemlighet mellan två parter har kommit att betyda mycket för datasäkerheten. Den viktigaste funktionen är att skapandet av den delade hemligheten kan ske helt över en osäker kommunikationskanal. Idén löste nyckeldistributionsproblemet som utgjorde den största svagheten i kryptering på den tiden och protokollet är i fortsatt användning än idag. Diffie-Hellman-nyckelutväxlingsprotokollet som Diffie och Hellmans idé utmynnade i har sin största nytta i den framåtsekretess som kan åstadkommas. Framåtsekretessen åstadkoms genom att för varje meddelande som sänds byta ut den delade hemligheten som används. Vad som åstadkoms genom detta är att om en attackerare lyckas hitta en delad hemlighet och därmed kan läsa kommunikationen har attackeraren endast tillgång till ett meddelande och inte kommunikationen i sin helhet.

Diffie-Hellman-nyckelutväxlingsprotokollet är trots den enkla aritmetik som protokollet bygger på svårt att implementera korrekt i praktiken. Svårigheterna med att åstadkomma en korrekt och säker implementation ligger främst i hur de primtal som används i nyckelutväxlingen skall väljas. Flera standarder och rekommendationer som utförligt behandlar det korrekta sättet att välja primtalen finns. Trots standarderna och rekommendationerna är det alltid människor som gör implementationerna som kommer att användas i praktiken. Den mänskliga faktorn ger upphov till att svagheter medvetet eller omedvetet smyger sig in i systemen.

Den mest betydande attacken mot en Diffie-Hellman implementation är Logjam attacken från 2015. I attacken utnyttjade ett forskarteam svagheter i TLS och exportklassade krypteringsalgoritmer för att förhållandevis effektivt kunna tjuvlyssna eller ändra på säkra kommunikationer. Logjam attacken bevisar hur den

mänskliga faktorn inom tekniken gör det möjligt att attackera något som matematisk kan anses säkert. Implementationer av Diffie-Hellman-algoritmen som använt svaga primtal eller ett stort antal implementationer som använder samma primtal är två misstag som uppdagades i forskarnas undersökning av Diffie-Hellmans användning i praktiken. Det fortsatta stödet för exportklassade krypteringsalgoritmer från 1990-talet innebär också en stor säkerhetsrisk. Att åtgärda de problem som forskarna upptäckte är relativt enkelt om man endast ser på Diffie-Hellman-algoritmens bristfälliga användning. De klienter som stöder Diffie-Hellman-nyckelutväxlingsprotokollet behöver sluta stöda små Diffie-Hellman-grupper och endast godkänna grupper med en storlek på 1024-bitar eller större. Implementationerna av Diffie-Hellman-protokollet behöver börja använda sig av flera primtal och inte endast återanvända samma primtal. Skapandet av nya primtal är dock en svår process som lätt kan leda till andra svagheter om de inte väljs rätt. Därför är denna svaghet svårare att åtgärda. Servrarna som har en TLS-tjänst behöver också stänga av stödet för exportklassade krypteringsalgoritmer för att kunna klassas som säkra. Konfigurationen av servrarna måste också åtgärdas. Servrarna behöver byta ut sin privata nyckel för varje ny kommunikation.

9 Slutsats

Avhandlingen har behandlat hur Diffie-Hellman-nyckelutväxlingsprotokollet fungerar och de attacktyper som kan användas mot det samt hur dessa har åtgärdats. Funktionen hos Diffie-Hellman-nyckelutväxlingsprotokollet bygger på det diskreta logaritmproblemet. Problemet används som en envägsfunktion för att skapa en delad hemlighet mellan två parter över en osäker kommunikationskanal. Det har inte uppdagats direkta svagheter i Diffie-Hellman-algoritmens matematiska del, men algoritmens brist på autentisering har åtgärdats. Algoritmer som kan användas för att försöka lösa det diskreta logaritmproblemet utgör även ett hot mot Diffie-Hellmans säkerhet. De största svagheterna som finns med Diffie-Hellman-protokollet har inget att göra med algoritmens matematiska del utan har introducerats genom okunskap, felaktiga implementationer eller medvetet försvagande av algoritmen.

Källor

[1] D. Adrian *et al*, "Imperfect forward secrecy," 12.10.2015.

[2] "X9.42 - 1998, Public Key Cryptography for The Financial Service
Industry :
Agreement of Symmetric Keys Using Diffie-Hellman and MQV
Algorithms ©,"American National Standards Institute, Maj 21,American National Standards Institute, "X9.42 - 1998, Public Key Cryptography for The Financial Service
Industry :
Agreement of Symmetric Keys Using Diffie-Hellman and MQV
Algorithms ©," 21.5.1998.

[3] E. B. Barker, D. Johnson and M. E. Smid, "Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography," 2007.

[4] blacksheepnetworks, "blacksheepnetworks," *Blacksheepnetworks*. Hämtad: 22.03.2018. URL: http://www.blacksheepnetworks.com/security/info/misc/handbook/images/08-04.jpg

[5] Carts David, "A review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols," *SANS Institute*, 5.11.2001.

[6] A. Commeine and I. Semaev, "An algorithm to solve the discrete logarithm problem with the number field sieve," *Public Key Cryptography - PKC 2006* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, sid. 174-190.

[7] W. Diffie and M. Hellman, "New directions in cryptography," *Tit*, vol. 22, *(6)*, sid. 644-654, 1976. Hittas: http://ieeexplore.ieee.org/document/1055638. DOI: 10.1109/TIT.1976.1055638.

[8] ENISA, "Algorithms, key size and parameters
report – 2014," November, 2014.

[9] "RFC 7836,"IETF Network Working Group, IETF Network Working Group, "RFC 7836," Mars, 2016.

[10] "RFC 5114,"IETF Network Working Group, IETF Network Working Group, "RFC 5114," Januari, 2008.

[11] "RFC 4306,"IETF Network Working Group, IETF Network Working Group, "RFC 4306," December 2005.

[12] "RFC 3526,"IETF Network Working Group, IETF Network Working Group, "RFC 3526," Maj, 2003.

[13] "RFC 2631,"IETF Network Working Group, IETF Network Working Group, "RFC 2631," Juni, 1999.

[14] J. Katz and Y. Lindell, *Introduction to Modern Cryptography : Principles and Protocols.* 2007 Hittas: http://ebookcentral.proquest.com/lib/[SITE_ID]/detail.action?docID=4742564.

[15] A. S. Khader and D. Lai, "Preventing man-in-the-middle attack in diffie-hellman key exchange protocol," 2015, Hittas: http://ieeexplore.ieee.org/document/7124683. DOI: 10.1109/ICT.2015.7124683.

[16] McCurley Kevin, "The Discrete Logarithm Problem," *Proceedings of Symposia in Applied Mathematics,* vol. 42, sid. 49-74, 1990. Hittas: http://www.mccurley.org/papers/dlog.pdf.

[17] Raymond Jean-Françis och Stiglic Anton, "Security Issues in the Diffie-Hellman Key Agreement protocol," *IEEE Transactions on Information Theory*, vol. 22, 2002. Hittas: http://instantlogic.net/publications/DiffieHellman.pdf.

[18] P. Rewagad and Y. Pawar, "Use of digital signature with diffie hellman key exchange and AES encryption algorithm to enhance data security in cloud computing," 2013, Hittas: http://ieeexplore.ieee.org/document/6524434. DOI: 10.1109/CSNT.2013.97.

[19] Rouse Margaret, "Secure Shell (SSH)," *WhatIs.Com*, Mars, 2016. Hämtad: 2.3.2018. URL: https://searchsecurity.techtarget.com/definition/Secure-Shell

[20] Rouse Margaret, "man-in-the-middle-attack (MitM)," *IoT Agenda*, December, 2015. Hämtad: 5.3.2018. URL: https://internetofthingsagenda.techtarget.com/definition/man-in-the-middle-attack-MitM

[21] Rouse Margaret, "digital signature," *WhatIs.Com*, November 5, 2014. Hämtad: 5.3.2018. URL: https://searchsecurity.techtarget.com/definition/digital-signature

[22] Rouse Margaret, "message authentication code (MAC)," *WhatIs.Com*, November, 2010. Hämtad: 2.4.2018. URL: https://searchsecurity.techtarget.com/definition/message-authentication-code-MAC

[23] Rouse Margaret, "Internet Key Exchange (IKE)," *WhatIs.Com*, Mars, 2009. Hämtad: 3.3.2018 URL: https://searchsecurity.techtarget.com/definition/Internet-Key-Exchange

[24] Rouse Margaret, "hashing," *WhatIs.Com*, September, 2005. Hämtad: 5.3.2018 URL: https://searchsqlserver.techtarget.com/definition/hashing

[25] "PKCS #3: Diffie-Hellman KeyAgreement Standard,"RSA Laboratories, 1 November, RSA Laboratories, "PKCS #3: Diffie-

Hellman KeyAgreement
Standard,", 1.11.1993.

[26] S. Singh, *The Code Book*. Fourth Estate, London, 2000.

[27] Spring Tom. (Mar 2,). *DROWN Flaw Illustrates Dangers of Intentionally Weak Crypto*. Hämtad: 21.3.2018. URL: https://threatpost.com/drown-flaw-illustrates-dangers-of-intentionally-weak-crypto/116555/.

[28] Valenta Luke *et al*, "Measuring small subgroup attacks against Diffie-Hellman," *NDSS Symposium 2017*, 2017. Hittas: https://eprint.iacr.org/2016/995.pdf.

[29] W. Weisstein Eric, "Group Order," *MathWorld--A Wolfram Web Resource.* Hämtad: 22.3.2018. URL: http://mathworld.wolfram.com/GroupOrder.html