## 9. Web services

På grund av internet tillväxt de senaste åren har de visat sig att simpla protokoll har underlättat för generisk kommunikation. Exempelvis har *HTTP* gjort det möjligt att en klient och server har kunnat prata med varandra över internet på ett enkelt sätt. Däremot är denna klassiska modell begränsad till en viss mängd typer av applikationer, och det är här webbtjänster kommer in.

En webbtjänst erbjuder en sorts infrastruktur för mer generell och strukturerad form av kommunikation mellan klient och server. Exempelvis kan en webbtjänst göra det möjligt för kommunikation mellan olika organisationer utan någon mänsklig interaktion. En webbtjänst kan därför ses som ett *tillägg* i en webbserver, utöver alla andra förfrågningar den kan hantera.I praktiken kan en webbtjänst för ett företag som Amazon vara erbjuda kunden att söka produkter och få information om dessa, eller logga in på deras hemsida.

XML (*Extensible Markup Language*) är ett läsbart textbaserat dataformat som kan användas av webbtjänster när de utbyter information. Då formatet är dynamiskt används det i t.ex. *SOAP*-protokollet. SOAP kapslar in data och skickar denna till mottagare med t.ex. HTTP eller SMTP. Fortsättning kan en webbservice kommunicera på olika sätt, huvudsakligen synkront och asynkront - blocking respektive non-blocking. Eventbaserad kommunikation kan också användas, vilket innebär att en klient registrerar sig som lyssnare av händelser. I ett sådant system behöver klienten inte hela tiden skicka förfrågningar, utan blir informerad på andra hållet när något inträffat. Eventbaserad kommunikation är extra viktigt när många användare kräver uppdatering av information, men när det inte är säkert när informationen blir tillgänglig.

REST (*Representational State Transfer*) är en alternativt gränssnitt till SOAP-protokollet som definierar hur resurser manipuleras av klienter med hjälp av enkla HTTP-operationer - *GET*, *PUT*, *DELETE* och *POST*. Varje resurs har en eget unikt id, *URL*, som operationerna utförs på. Fortsättningsvis definierar REST att varje förfrågan för en resurs ska förse hela den resursen, och inte delar av den (då detta skulle kräva ytterligare förfrågningar av klienten för att få hela resursen). REST har på många ställen ersatt det äldre protokollet SOAP, men de används på flera ställen i olika grad.

URI i en webbtjänst-modell kan jämföras med remote object referenser i en distribuerad objektmodell. I båda modellerna pekar en unik identifierare ut en resurs som operationer kan utföras på. Däremot är den viktiga distinktionen att en webbtjänst inte får skapa ett nytt remote object, utan agerar endast på tillgängliga resurser. Fortsättningsvis kan webbtjänster jämföras med den äldre tekniken CORBA. De är lika på flera fronter i att datorer kommunicerar med varandra, till och med via olika organisationer. Däremot är den främst skillnaden deras tänkta användsområde, och hur de designades efter detta. CORBA var endast tänkt att användas inom en organisation och faller därmed i sin struktur när tekniken används på en större skala - problem med namn, referenser och simplicitet.

En tjänstbeskrivning (*service description*) formar basen hur en klient och server pratar med varandra. Detta kan vara bland annat vilken typ av meddelande som ska skickas (såsom meddelandet format). För webbtjänster används *WSDL* (*Web Service Description Language*) för deras tjänstbeskrivning som definierar bland annat vilka komponent som ingår i deras kommunikation - typer (boolean, int..), meddelanden, gränssnitt.

Säkerheten för meddelanden en webbtjänst utbyter kan i många fall vara viktig, t.ex. om en webbtjänst skickar meddelanden innehållande personlig information. För XML kan dedikerade taggar användas som markerar var krypterad information och signaturer börjar och slutar. För en XML-signatur krävs olika algoritmer, exempelvis SHA-1 för meddelande digest och DSA/RSA för signaturen. Fortsättningsvis används 3DES, AES-128 och AES-256 för block cipher och bland annat 3DES för att kapsla in den symmetriska nyckeln. Diffie hellman algoritmen används för nyckelavtalet.

## 10. Peer-to-peer Systems

Peer-to-peer (P2P) är ett sätt att kommunicera för eliminera begränsningar och krav på mellanliggande servrar och deras infrastruktur. Traditionella klient-server-system baseras på en eller flera nära samarbetande servrar (serverkluster). Ett sådant systems hårdvara beror helt och hållet på dess hårdvara och nätverksuppkopplingen till det. I ett P2P-system däremot delas resurser från samtliga datorer i ett nätverk, och är därmed helt och hållet decentraliserat. Ett sådant system skalas mer dynamiskt då lasten delas på datorerna som är kopplade till nätverket - varje användare i system bidrar till systemet. P2P-system möjliggör distribuerade beräkningar genom att låta en stor mängd datorer som inte gör någonting i stunden att köra bakgrundsprocesser.

För att lokalisera noder och objekt i ett peer-to-peer-system används en distribuerad algoritm som kallas *routing overlay*. Detta kallas för en overlay på grund av att den agerar som ett mellanlager för en klient och dess anslutning till nätet - algoritmen utförs på applikationslagret “ovanpå” nätverkslagrets egna routing. Fortsättningsvis är det vanligt i ett P2P-system att resurser lagras på flera ställen (på flera olika noder i nätverket) för att förbättra dess tillgänglighet (då noder när som helst kan koppla sig ifrån nätverket). Routing overlay ser då även till att hålla reda på var alla dessa resurser finns någonstans, genom att lagra och hålla reda på deras globala unika id *GUID (Globally Unique Identifier)*, samt den effektivaste vägen dit. Routing overlay är alltså ett sätt att underlätta för noder i ett peer-to-peer-system att samarbete med alla noder som är kopplade i nätverk både vad gäller kommunikation och resursdelning.

Fördelarna med ett P2P-system är sammanfattningsvis möjligheten att samarbeta med många andra datorer utan någon centraliserad server - en dator som inte utför något för stunden kan exempelvis istället användas för att dela resurser till andra som kräver dem. System är också väldigt skalbart genom att en dator både belastar och avlastar nätverket. Ett P2P-system faller dock på hur effektivt data kan muteras. På grund av att resurser kan finns på flera noder samtidigt måste alla uppdatera sin egen kopia. Det är kostsamt både i form av mängden skrivningar som måste ske, men också för att se till att resursen inte efterfrågas under tiden modifieringen sker.

## 11. Security

I ett distribuerat system finns det krav på olika håll för säkerhet. Bland annat när personliga uppgifter hanteras, exempelvis när bankkontonummer eller personnummer skickas. Då varje sådan resurs kan hanteras i olika processer runt om i ett distribuerat system är det viktigt att säkra upp den processen och se till att den kommunicera på säkra kanaler till andra processer. En *säkerhetsmekanism* är vad som driver igenom *säkerhetspolicies*.

Översiktligt är målet med säkerhet inom ett distribuerat system att motverka *läckage*, *manipulering* *av data* och *vandalism*. Fortsättningsvis kan säkerhetsattacker på ett system delas in i olika typer. Bland de vanligaste typerna hör *eavesdropping*, *masquerading* och *denial of service (DoS)*. Att se över dessa aspekterna i ett system är särskilt viktigt för just *transaktioner* som t.ex. handla på internet, e-post och skicka pengar genom banken. När ett säkert distribuerat system ska designas är det viktigt att tänka på det värsta fallet. Detta innebär bland annat att se nätverk som helhet som osäkra - meddelanden kan falsifieras, ip-adresser kan bli *spoofade* osv.

Kryptering är en process som inom distribuerade system används när en sändare och en mottagare ska kommunicera på ett säkert sätt. Sändare vill vara säker på att det är endast mottagaren som kan öppna och läsa det skickade meddelandet. All form av kryptering är baserad på *nycklar*. Idéen är att ett meddelande inte kan läsas av någon annan än den med en nyckel. Fortsättningsvis finns det två huvudtyper av kryptering som har både för- och nackdelar. Den första är *symmetrisk kryptering*, och går ut på att sändare och mottagare delar en hemlig nyckel som de använder för att skicka meddelanden. Denna nyckel får inte delas med någon annan part.

Den andra typen är *asymmetrisk kryptering*, och går ut på att ha en publik och en hemlig nyckel (kallas därför ibland för *public key encryption*). Den publika nyckeln delas ut till vem som helst som vill kommunicera med den som äger den hemliga nyckeln (endast den hemliga nyckeln kan dekryptera vad de publika nycklarna krypterat). Det viktiga med asymmetrisk kryptering är att hålla den hemliga nyckeln privat. Kryptering användes således för att upprätthålla integritet och sekretess.

Vidare implementeras *digitala signaturer* med hjälp av asymmetrisk kryptering. Detta går ut på att kryptera ett meddelande med den hemliga nyckeln och skicka det till mottagare. Alla med en korresponderande publik nyckel kan dekryptera detta och därmed verifiera att sändaren måste vara ägaren till den privata nyckeln. En digital signatur är menad att uppnå samma nivå av verifiering som en vanlig signatur - autencitet, integritet och oförnekbar. Digitala signaturer går också att uppnå med symmetrisk kryptering med hjälp av MAC (*Message Authentication Codes*).

Ett annat användningsområde för just asymmetrisk kryptering är nyckelutbyte som ofta sker inledningsvis till säker kommunikation. *Diffie-Hellman nyckelöverföring* syftar på en teknik som på ett säkert sätt gör att två parter kan överföra nycklar till varandra. Tekniken går ut på att använda väldigt stora tal tillsammans med den matematiska modulus operationen.

Innan kryptering sker används ibland en s.k. *secure hash function* vars uppgift är att omvandla ett meddelande till ett rad bytes med förutbestämd längd, en *digest*. Att ha en bestämd längd på ett meddelande kan snabba upp krypteringen. En hashfunktion är också menat till att vara en envägsoperation, vilket innebär att det inte går att veta hur ett meddelande såg ut bara genom att undersöka dess digest (något som är användbart när t.ex. lösenord ska lagras i en databas). Några exempel på hashfunktioner är MD5, SHA-1 och SHA-256.

I praktiken var *Needham-Schroeder* autentiseringsprotokollet det tidigaste praktiska säkerhets protokollet. På detta protokoll har det populära *Kerberos* baserats, och är ett generellt autentiserings- och säkerhetsprotokoll. Översiktligt hanterar Kerberos tre typer av säkerhetsobjekt:

* Ticket  
  Agerar som en sorts biljett till en specifik server och verifierar att användare nyligen blivit autentiserad av Kerberos.
* Autentiserare  
  En token som skapats av klienten och skickas till servern för att bevisa sin identitet.
* Sessionsnyckel  
  En hemlig nyckel som skapats av servern och skickas till klienten. Klienten använder denna för att säkert kommunicera med servern i just den sessionen.

## Reflektion

Under projektet i årskurs 2 utvecklade vi ett webbgränssnitt, samt ett webbsida. Webbgränssnittet var byggd på principen REST, något som tog lite tid att vänja sig vid. Den främsta utmaningen var hur vi utvecklar ett effektivt API när det måste vara *stateless* - hur hanterar vi cache?

Det som skulle vara intressant med just vår uppgift var ifall vi fick ta hänsyn till autentisering, då projektet saknade detta (API:et var helt öppet via en URL). Troligen hade vi behövt vänja oss vid olika protokoll som används på webben, då vårt företag befann sig i Microsofts ekosystem - SSO? Kerberos?