IDATT2503 Cryptography

Public key Cryptography

Lecture 4, November 1.

Dag Olav Kjellemo

Plan

- Public key cryptography
- Some background number theory
- RSA
- Diffie-Hellman key exchange
- ElGamal – not details
- Digital signatures
- Examples of protocols

Summary so far

Symmetric ciphers – requires common secret shared between sender and receiver.

Can be used for

- Secrecy (symmetric encryption, e.g. AES)
- Integrity (secure hash functions)
- Authenticity (MAC’s)

A major challenge in many use cases is the

Key distribution problem

Public key cryptography

Also called asymmetric cryptography.

- Addresses the key distribution problem
- Two keys, one public to encrypt with, another private to decrypt with (so, asymmetric).
- Security based on one-way functions:
- Easy to calculate one way, but hard to go other way
- Examples:
  - RSA, Diffie-Hellmann, ElGamal
- Provides "computational security“
  - Secret key is mathematically possible to work out from public key, just hard to work out without the secret information.

Public key secrecy

- There can be no perfect secrecy in public key cryptography. All information is available, but computationally hard to actually use in an attack.
- With sufficient computational resources, PK cryptosystems can be completely broken = find private key, without knowing any messages.
- For private key cryptography, the key is secret, and only using information from its use, can one infer information about the messages and the key.

Number theory

- Modular arithmetic, the quotient-residual theorem
- Prime numbers, greatest common divisor, relatively prime numbers
- Euclidean extended algorithm
- Fermat’s Little Theorem
- Euler's Totient Function and Eulers theorem
- The order of a number modulo p
- Efficient calculation of high powers using repeated squaring
- Chinese remainder theorem

Quotient-remainder theorem

For integers n, a, a > 0, there are uniquely defined integers q and r, such that

$$n = qa + r, 0 \leq r < a$$

We call q the quotient, r the remainder.

There is also a version for polynomials over a field,

Then the degree of the remainder is smaller than the degree of the a

In the same way that we work modulo n in $\mathbb{Z}_n$ we can work modulo a polynomial p. With $p(x) = x^8 + x^4 + x^3 + x + x$ this gives us the Galois field $GF(2^8)$

Modular arithmetic

- $\mathbb{Z}_n$ is the set of integers modulo n, usually just written as 0,1,2,…,n-1,
- Multiplicative inverse of a mod n: $a^{-1}a \equiv 1 \ (mod\ n)$. Exists for all a relatively prime to n, i.e. gcd(a,n)=1
- Multiplying by a^{-1} is the same as dividing by a.
- $(a^{-1})^{-1}= a$, so inverse of inverse is itself
  - $3^{-1} = 7$ mod 20, since $7 \cdot 3 \equiv 1 \ (mod\ 20)$
  - Usually found found by extended euclidean alg,
  - Also, if $a^m \equiv 1$, then $a^{-1} \equiv a^{m-1}$

Some example calculations

Euclidean algorithm

Definition

Hvis r = 0, så er gcd(b,r) = b, og vi har

$$\gcd(a, b) = b$$

Hvis r>0, bruker nå kvotient-rest på b og r:

$$b = q_2 r + r_2$$

Her er $0 \leq r_2 < r <$ b.

Nå gjentar vi hele prosessen, nå med r_1 og r_2.

Så lenge resten er større enn 0, så fortsetter vi. Når resten er 0, la oss si etter k trinn, dvs. $r_k = 0$, så har vi at

$$\gcd(r_{k-1}, 0) = \mathrm{r_{k-1}}$$

Vi setter sammen hele kjeden med likheter og får

$$\gcd(a, b) = \gcd(b, r_1)$$

$$= \gcd(r_1, r_2) = \cdots = \gcd(r_{k-1}, 0) = r_{k-1}$$

Example

Multiplicative inverses, linear diophantine equations, and extended euclidean algorithm

Let's write about what

$$ab \equiv 1 \pmod{n}$$

means:

$$n \mid (ab - 1)$$
$$ab - 1 = kn$$
$$ab - kn = 1$$
$$ab + (-k)n = 1$$

So given a and n, we can find integers b and k such that $ab + (-k)n = 1$?

We have a Diophantine equation that has a solution when a and n are multiplicative inverses of each other.

An equation where we only want integer solutions is called a Diophantine equation.

Euclid's extended algorithm

- Finding an integer solution to the equation

$$ax + ny = d$$

- For d = 1, we also find the multiplicative inverse of a modulo n

- Example:

Chinese remainder theorem

- Let m, n be relatively prime integers, and a, b integers. Then there is exactly one solution modulo mn to the set of equations

$$x \equiv a \ (mod\ m)$$
$$x \equiv b \ (mod\ n)$$

Example: m = 5, n = 6, a = 2, b = 3:

Integers satisfying $x \equiv 2 \ (mod\ 5)$ are: 2, 7, 12, 17, 22, 27 …

Integers satisfying $x \equiv 3 \ (mod\ 6)$ are: 3, 9, 15, 21, 27,

27 is only solution between 0 og mn = 30.

Chinese remainer theorem proof

General solution: Since m and n are relatively prime, we can find integers u og v from Euclid's extended algorithm such that $um + vn = 1$.

Then we get a solution

$$x = vna + umb$$

In the example above, we have $(-1) \cdot 5 + 1 \cdot 6 = 1$, so $u = -1, v = 1$ giving

$$\begin{aligned} x &= 1 \cdot 6 \cdot 2 - 1 \cdot 5 \cdot 3 \\ &= -3 \\ &\equiv -3 + 30 \equiv 27 \ (mod\ 30) \end{aligned}$$

In case that $a = b$, we get

$$x = (vn + um)a = a$$

This is however quite obvious

Note: The formula gives an integer solution.

RSA – textbook version

Alice creates two keys, a public key e for encryption, and a private key d for decryption.

1. Select two (large) primes p and q randomly and calculate n=pq.
2. Choose integer e relatively prime to $(p-1)(q-1)$.
3. Calculate the multiplicative inverse d to e modulo $(p-1)(q-1), so$

$$de \equiv 1\ (mod\ (p-1)(q-1)$$

1. Publish (n,e) as public key
2. Keep p,q,d private

RSA – Encryption and decryption

If Bob wants to send Alice a message M (coded as numbers – note that M must be smaller than pq, it must be split up), he sends the encrypted message C calculated as follows:

$$C \equiv M^e \pmod{n}$$

Alice can now decrypt C using her private key:

$$M \equiv C^d \pmod{n}.$$

RSA Example

$p = 7$ og q =13.

$n = 91$, $(7-1)(13-1) = 72$. Here we can choose eg. $e = 5$.

- We calculate $d = 29$

  We can check: $5 \cdot 29 = 145 = 2 \cdot 72 + 1 \equiv 1 \pmod{72}$.

  Public key is (5,91), while private key is (29, 91= 7 x 13).

- Bob will send the message 9, calculating

$$9^5 = 59049 \equiv 81 (\mathrm{mod}\ 91)$$

- He sends 81 to Alice. Alice decrypts this as

$$81^{29} \equiv 9 \pmod{91}.$$

Why RSA works

Fermat's Little theorem: $x^p \equiv p\ (mod\ p)$, for any x

Since $ed \equiv 1\ (mod\ (p-1)(q-1))$, we get

$$ed = k(p-1)(q-1) + 1$$

Then

$$x^{ed} \equiv (x^{p-1})^{k(q-1)} \cdot x \equiv x\ (mod\ p)$$

Likewise for q,

$$x^{ed} \equiv x\ (mod\ q)$$

This gives

$$x^{ed} \equiv x\ (mod\ pq)$$

(This can also seen directly from properties of primes and divisibility)

Efficient calculation of high powers mod n

Når vi regner ut høye potenser, så kan vi gjenbruke tidligere utregnede potenser. En måte er å bruke *gjentatt kvadrering*

Eksempel: Regn ut $5^{117} \bmod 209$

Først, skriv 117 som sum av potenser av 2:

$$117 = 2^0 + 2^2 + 2^4 + 2^5 + 2^6$$

Potensregler gir oss følgene:

$$5^{117} = 5^{(2^0+2^2+2^4+2^5+2^6)}$$
$$= 5^{2^0} \cdot 5^{2^2} \cdot 5^{2^4} \cdot 5^{2^5} \cdot 5^{2^6}$$

5^{2^0}	5	5
5^{2^1}	$5^2 \equiv 25$	
5^{2^2}	$25^2 \equiv -2$	$-2 \cdot 5 \equiv 199$
5^{2^3}	$(-2)^2 \equiv 4$	
5^{2^4}	$4^2 \equiv 16$	$16 \cdot 199 \equiv 49$
5^{2^5}	$(16)^2 \equiv 47$	$47 \cdot 49 \equiv 4$
5^{2^6}	$47^2 \equiv 119$	$119 \cdot 4 \equiv 58$

Security of RSA

RSA is secure since it is hard to factorize n as far as we know (but not for quantum computers). Too many to check all, multiplication hides well the factors.

There are many primes, and relatively easy to check if a number is prime (primality testing).

For RSA to be secure, it is important to ensure that

- p and q are properly randomized. If not, there are less primes an adversary need to consider in an attack.
- p and q should not be close together.
- p-1 and q-1 should have no small prime factors

Fermats factorization method

For p and q close together, there is an efficient way to find the factors of n=pq, p and q odd. Assume p < q

a = (p+q)/2 and b = (q-p)/2 are integers, and

$$p = a - b, q = a + b$$

This gives $pq = (a-b)(a+b) = a^2 - b^2$

If p and q are close, then b is small.

1. Let $a = \lceil \sqrt{n} \rceil$
2. Calculate $b = \sqrt{a^2 - n}$.
3. If b is an integer, then $n = (a-b)(a+b)$, else

   If a > b, let a = a+1 and repeat from step 2, else

   n is a prime

Attacking RSA: Pollard (p-1)

En angriper kjenner produktet $n = pq$, men ikke faktorene p og q. For en a og en u kan han regne ut $a^u \bmod n$. Han ønsker at $a^u \equiv 1 \pmod p$, for da vet han at $p \mid a^u - 1$, og siden $p \mid n$ også, så må $p \mid \gcd(a^u - 1, n)$. Dette kan han regne ut effektivt ved Euklids algoritme.

Hvilke *u* som fungerer?

Det vil fungere med et tall u slik at $(p-1) \mid u$, dvs. $u = (p-1)k$. For da vil

$$a^u = a^{(p-1)k} = (a^{p-1})^k \equiv 1^k \equiv 1 \pmod p)$$

Her har vi brukt

Teorem (Fermats lille teorem)

For primtall p og heltall a slik at $p \nmid a$, så er

$$a^{p-1} \equiv 1(\mod p)$$

NTNU

Pollard $p-1$

Oppsummert: En angriper

- Gjetter på en B
- Regner ut $A = a^{B!} \pmod n$
- Regner ut $\gcd(A-1, n) = F$. Hvis $F > 1$ så er den en av primtallsfaktorene i n.

Hvor stor må B være?

Hvis vi primtallsfaktoriserer $p-1$,

$$p - 1 = q_1^{t_1} \dots q_r^{t_r} = Q_1 \dots Q_r$$

så ser vi at hvis vi velger B til å være det høyeste av $Q_1, \dots Q_r$, så vil $p-1 \mid B!$ (fakultet).

For at angrepet skal være effektivt, må en av Q_i-ene være relativt små.

Det holder å finne det ene primtallet, så effektiviteten av angrepet avhenger av den minst primtalls-potensen til p eller q.

Problems with textbook RSA

Even if we have chosen p,q,d,e properly, there are problems with RSA used alone.

- Plaintext is always encrypted to the same ciphertext
- Small
- Eva can easily tamper with a ciphertext: Even if she does not know x, she can do certain manipulations to it, eg. Multiplying it with 2:
  - If $y = x^e$ is a cipher text, so is $y' = 2^e \cdot x^e \ (mod\ n)$.
- The recipient decrypts and receives the message 2*x*, without knowing that it has been tampered with.

RSA in real life

- Randomized padding to avoid equal messages being encrypted the same each time
- Using a MAC to avoid tampering
- Strong cryptographic randomization in choice of p and q

Diskrete logarithm problem, Diffie-Hellmann key exchange

Order of an element in $\mathbb{Z}_{\boldsymbol{n}}^{*}$

- $\mathbb{Z}_{\boldsymbol{n}}^{*}$ is the set of elements in $\mathbb{Z}_{\boldsymbol{n}}$ that have a multiplicative inverse. Its the numbers relatively prime to n
- This is not "just a quantity." It has multiplication and is what is called a group during multiplication. (In this context, we do not use the addition)
- *Orde* of $\mathbb{Z}_{\boldsymbol{n}}^{*}$ is the number of elements in the quantity. Examples:
- Given an element a in $\mathbb{Z}_{\boldsymbol{n}}^{*}$, Then we can look at the sequence
$$a = a^1, a^2, a^3, \ldots$$
- This one has to repeat itself. Example a = 2, n = 11

  2,4,8,5,10,9,7,3,6,1,2,4,…
- The order of an element a in $\mathbb{Z}_{\boldsymbol{n}}^{*}$ is the smallest power of a giving 1, $a^k \equiv 1 \ (mod\ n)$

NTNU

Discrete logarithm problem

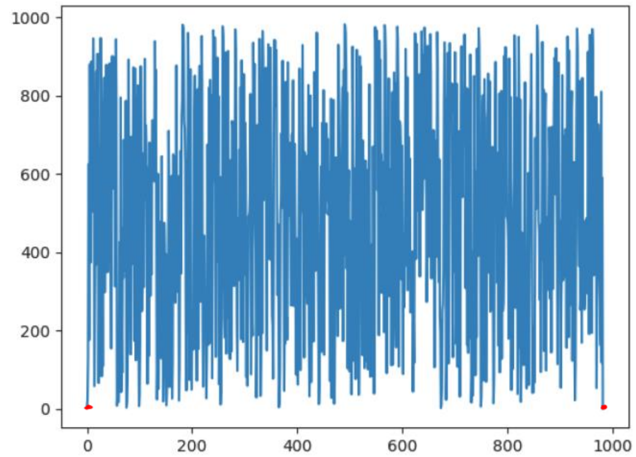
The powers $a^k, k = 1,2,3, \ldots$ turn out to be unpredictable

See example on next slide

Its in general computationally hard to find k, given a, a^k modulo n.

Eksponensialfunksjoner mod n og diskrete logaritmer

Når vi regner høye mod n, så blir verdiene uforutsigbare. Dette kan vi utnytte!



Figur: Plot av verdiene $5^x \bmod 983$ for $x = 1, 2, ..., 982$

Det diskrete logaritmeproblemet

- $f(x) = \alpha^x \bmod p$ er (diskrete) *eksponensialfunksjoner*.
- De inverse funksjonene til disse kalles **diskrete logaritmer** mod p (med base α).
- **Det diskrete logaritmeproblemet** er å finne eksponenten

  $$k = \log_\alpha \alpha^k \bmod p$$

  når vi kun kjenner $\alpha^k \bmod p$.
- Dette er generelt vanskelig = beregningskrevende: Det er ingen kjent effektiv måte å finne dem.
- Danner basis for Diffie-Hellmann nøkkelutveksling og ElGamal kryptering.

Potenser, logaritmer og primitive elementer

Fra Fermats lille teorem så vet vi at

$$\alpha^{p-1} \equiv 1 \pmod{p} \text{ når } p \nmid \alpha$$

Finnes det noen mindre (positiv) eksponent e slik at $\alpha^e \equiv 1 \pmod{p}$?

Definisjon (Orden til element, Primitive elementer)

- *Det minste positive talle e slik at* $\alpha^e \equiv 1 \pmod{p}$ *kalles* ***ordenen*** *til* α *i* $\mathbb{Z}_p^*$.
- *Hvis ordenen er lik* $p-1$, *så kalles* α *for* ***et primitivt element*** *i* $\mathbb{Z}_p^*$

Det er et teorem at ordenen deler $p-1$.

Diskrete logaritmer - Formelt

Definisjon (Diskret logaritme)

Hvis $\alpha^x = y \pmod{p}$, så sier vi at x er ***den diskrete α-logaritmen*** *til y i $\mathbb{Z}_p{}^*$ og skriver $x = \log_\alpha y$.*

Eksempler:

- $log_3(5) = 5$ i $\mathbb{Z}_7^*$ fordi $3^5 \bmod 7 = 5$
- $log_2(5)$ er ikke definert i $\mathbb{Z}_7^*$ fordi ingen potens av 2 er 5 (mod ~~5~~ 7).
- $\log_5(367) = 904$ i $\mathbb{Z}_{983}^*$ fordi $5^{904} \bmod 983 = 367$

Definisjon (Diskrete logaritme-problemet)

Gitt $\alpha, \beta \in \mathbb{Z}_p^*$*, finn a mellom 0 og p slik at* $\alpha^a = \beta$ *som elementer i* $\mathbb{Z}_p^*$

- Det diskrete logaritmeproblemet (DLP) er kryptografisk nyttig fordi det er (antatt) vanskelig å løse, samtidig som eksponensiering er relativt effektiv operasjon.
- Flere kryptografiske systemer er basert på DLP, som Diffie-Hellmann nøkkelutveksling, ElGamal, og andre varianter.
- Det samme problemet finnes også i andre sykliske grupper enn $\mathbb{Z}_p^*$.
- En syklisk gruppe er et mengde med en operasjon, og hvor det finnes primitive elementer.
- Et eksempel er såkalte elliptiske kurver, hvor en har definert en operasjon (addisjon) på punktene. En kan formulere et diskret logaritmeproblem her på disse, og blir da hetene *elliptisk kurve diskret logaritme-problem*.

Diffie-Hellmann nøkkelutveksling

- **Diffie-Hellmann key exchange** var av de første nøkkelutvekslingsalgoritmene som kom (sammen med konseptet asymmetrisk kryptografi) og viste at det var mulig med sikker nøkkelutveksling over åpne linjer.

Diffie-Hellmann nøkkelutveksling

- Protokollen er som følger når Alice og Bob vil utveksle nøkler (utregninger $\bmod p$):
  1. Først blir de enige om et stort primtall p, og et heltall n. Disse tallene utveksler de åpent.
  2. Deretter velger Alice et tall a og Bob et tall b som de holder for seg selv.
  3. Alice sender $n^a = a_1$ til Bob, Bob sender $n^b = b_1$ til Alice, åpent.
  4. Alice regner ut $k = b_1^a$, og Bob regner ut $k = a_1^b$. **Dette er deres felles nøkkel**.

Potensregneregler sier oss at

$$b_1^a = (n^b)^a = n^{ab} = (n^a)^b = a_1^b$$

Angrep på ElGamal: Shanks algoritme

- Et brute force angrep er å regne ut potenser α^k til vi treffer β.
- Shanks algoritme gjør en “time-memory” trade-off (mindre utregninger men med større minne-behov). Vi genererer to lister med lengder $m = \sqrt{p}$, og ser etter en verdi som forekommer i begge listene.
- Dette kan vi vise at har tidskompleksitet og lagringskompleksitet $\sqrt{p}$

Algoritme (Shanks algoritme)

Input: p, α, β

- $m \leftarrow \lceil \sqrt{p} \rceil$
- *for* j *fra 0 to* $m-1$*: Beregn* α^{mj}
- *Hash eller sorter parene* (j, α^{mj}) *for effektive oppslag på andre koordinat.*
- *for* i *fra 0 til* $m-1$*: Bergen* $\beta\alpha^{-i}$
- *Hash eller sorter parene* (j, α^{mj}) *for effektive oppslag på andre koordinat.*
- *Finn par fra de to listene med like andre-koordinat*
- $\log_\alpha \beta = (mj + i) \bmod p$

Hvorfor virker Shanks algoritme

- At det er en løsning er enkelt: Hvis $\alpha^{mj} = \beta\alpha^{-i}$ så ganger vi bare med α^i på begge sider:
$$\beta = \alpha^{mj+i}$$
og vi har $a = mj + i$.
- At vi finner en løsning er basert på at
$$\log_\alpha \beta = mj + i, \quad 0 \leq i < m$$
ved kvotient-rest-teoremet. Her er $m \leq \lceil\sqrt{p}\rceil$. Da må $0 \leq j < m$, for ellers ville $mj \geq (\lceil\sqrt{p}\rceil)^2 \geq p$, og det ville motstride at $\log_\alpha \beta < p - 1$ ved Fermats lille teorem.

Shanks algoritme eksempel

Bruk Shanks algoritme til å angripe systemet fra tidligere, med

$$p = 29,\ \alpha = 11,\ \beta = 16$$

Digital signatures

- What exactly is the point of "regular" signatures?
- They link a person and a document together, in a way that is (hopefully) difficult/impossible to forge and deny.
- So do digital signatures, only that they make it algorithmic.
- In other words, they must be linked both to a document and to a person and should be difficult/impossible to forge and deny.
- "Regular" signatures are verified by comparing to other signatures, digital signatures come with their own verification algorithm.
- However, digital signatures have a problem that "regular" signatures don't have: A copy is identical to the original. This means that we must take measures to prevent unauthorized reuse.
- This means that security is typically based on the fact that it is computationally practically impossible to forge a digital signature, not that it is actually impossible.

Definition of digital signature system

- $\mathcal{P}$ is set of possible messages
- $\mathcal{A}$ is set of possible signatures
- $\mathcal{K}$ is set of keys

For each k in $\mathcal{K}$, there is a signing and algorithm and a verification algorithm:

$$sig_k: \mathcal{P} \rightarrow \mathcal{A}$$
$$ver_k: \mathcal{P} \times \mathcal{A} \rightarrow \{true, false\}$$

Such that

$$ver_k(x, y)$$

Is True if and only if $y = sig_k(x)$. We say then that the signed pair (x,y) is valid.

Comments to signatures

- Both the signing and verification algorithm should run in polynomial time.
- If Eve manages to produce a valid pair (x, y) not from Alice, then we say that y is a forgery (of the signature)

Many of the same attacks on MACs are relevant to signatures.

Attack models for digital signatures

- Only known key attack
- Known message attack: Oscar (the attacker/adversary) has access to previous valid pairs, but has not chosen which
- Chosen message attack: Oscar has Alice sign messages chosen by Oascar

The goals of the attacker can be:

- Total break: Oscar obtains the key for signing
- Selctive forgery: Oscar can forge signature of a given new message x with some non-negligible probability
- Existential forgery: Oscar can produce valid pair (x,y) for some x with a non-negligible probability.
- Oscar replaces Alice’s signature with his own.

Digital signatures with RSA

- RSA can be used for digital signing as follows:
- Alice wants to sign a message x. Then she uses the private decryption function to sign.
- The verification feature is then her public key – anyone can use it and compare the result with the original message.

Digital signatures with RSA

RSA can be used for digital signing. Alice wants to sign x. She then

- Calculates h(x) with a secure hash function.
- She “decrypts) h(x) with her ***private*** RSA key.
- This y = d(h(x)) is the signature to x

Verification is then done by anyone:

- “Encrypt” y with Alice’s public key, giving e(y)
- Calculate hash(x).
- If hash(x) = e(y), the signature is valid signature by Alice.

Problems with signature without hashing first

Existential forgery is easy.

- Oscar chooses any y, and uses Alice's public key to «encrypt» it.
- Then (e(y), y) is valid pair, since we verify it by checking that e(y) = e(y)

If message is first hashed, when Oscar chooses y, he has no control of what e(y) is, and if hash is pre-image resistant, then cannot find some x with hash(x) = e(y)

Combining public key encryption with signature: UPDATE

Encrypt and sign, or the other way around?

Whats best depends on the usage/situation, there are strengths and weaknesses for each

We will not go into details, but One solution is to use hybrid encryption:

- Exchange keys using asymmetrical crypto (RSA, Diffie-Hellmann)
- Use this key to authenticate and encrypt message.

RSA in reality

Some advice in practical world

- Do not depend on your own solutions when it comes to security
- Check what standards are used for the use case you have at hand.
- Searching the web may give you opinonated and outdated answers.