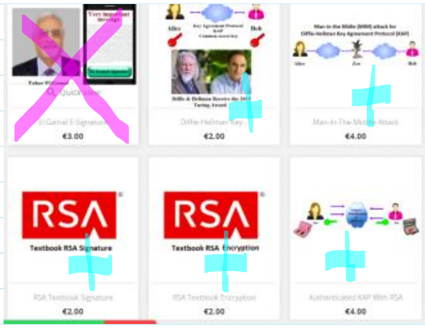
111_007 PKCS Scnorr-Id-Sig_AKAP

Midtermexam will be held in 31-st of October during practice lecture.

https://imimsociety.net/en/14-cryptography



El Gamal E-Signature €3.00

Diffie-Hellman Key... €2.00

Man-In-The-Middle Attack €4.00

RSA Textbook Signature €2.00

RSA Textbook Encryption €2.00

Authenticated KAP With RSA €4.00

Quick view
Textbook RSA Signature
Textbook RSA Encryption
Alice
Bob
Zoe
Diffie & Hellman Receive the 2015 Turing Award
Man-in-the-Middle (MiM) attack for Diffie-Hellman Key Agreement Protocol (KAP)

Cryptography: Information Confidentiality, Integrity, Authenticity and Authority (Person Identication).

Symmetric cryptography ------------- Asymmetric cryptography 1976 Public Key CryptoSystems - PKCS

Symmetric encryption: block ciphers stream ciphers	Asymmetric encryption E-signature - Public Key Infrastructure - PKI E-money
H-functions, Message digest	E-voting
HMAC H-Message Authentication Code	Digital Rights Management - DRM: Marlin Etc.

Public Key CryptoSystems - PKCS

1. Principles of Public Key Cryptography

Instead of using single symmetric key shared in advance by the parties for realization of symmetric cryptography, asymmetric cryptography uses two *mathematically* related keys named as private key and public key we denote by **PrK** and **PuK** respectively.

PrK is a secret key owned personally by every user of cryptosystem and must be kept secretly. Due to the great importance of **PrK** secrecy for information security we labeled it in red color. **PuK** is a non-secret personal key and it is known for every user of cryptosystem and therefore we labeled it by green color. The loss of **PrK** causes a dramatic consequences comparable with those as losing password or pin code. This means that cryptographic identity of the user is lost. Then, for example, if user has no copy of **PrK** he get no access to his bank account. Moreover his cryptocurrencies are lost forever. If **PrK** is got into the wrong hands, e.g. into adversary hands, then it reveals a way to impersonate the user. Since user's **PuK** is known for everybody then adversary knows his key pair

(**PrK**, **Puk**) and can forge his Digital Signature, decrypt messages, get access to the data available to the user (bank account or cryptocurrency account) and etc.

Let function relating key pair (**PrK**, **Puk**) be ***F***. Then in most cases of our study (if not declared opposite) this relation is expressed in the following way:

PuK=***F***(**PrK**).

In open cryptography according to Kerchoff principle function ***F*** must be known to all users of cryptosystem while security is achieved by secrecy of cryptographic keys. To be more precise to compute **PuK** in function ***F*** must be defined using some parameters named as public parameters we denote by **PP** and color in blue that should be defined at the first step of cryptosystem creation. Since we will start from the cryptosystems based on discrete exponent function then these public parameters are

PP = (***p***, ***g***).

Notice that relation represents very important cause and consequence relation we name as the direct relation: when given **PrK** we compute **PuK**.

Let us imagine that for given ***F*** we can find the inverse relation to compute **PrK** when **PuK** is given. Abstractly this relation can be represented by the inverse function $\boldsymbol{F^{-1}}$. Then

PrK=$\boldsymbol{F^{-1}}$(**PuK**).

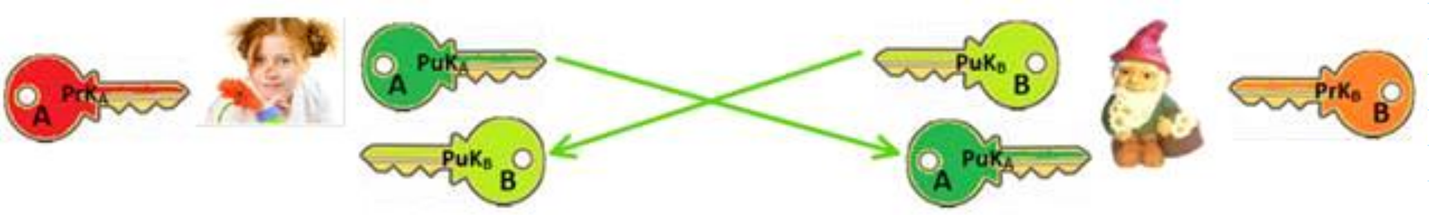
In this case the secrecy of **PrK** is lost with all negative consequences above. To avoid these undesirable consequences function ***F*** must be **one-way function** – OWF. In this case informally OWF is defined in the following way:

1. The computation of its direct value **PuK** when **PrK** and ***F*** in are given is effective.
2. The computation of its inverse value **PrK** when **PuK** and ***F*** are given is infeasible, meaning that to find $\boldsymbol{F^{-1}}$ is infeasible.

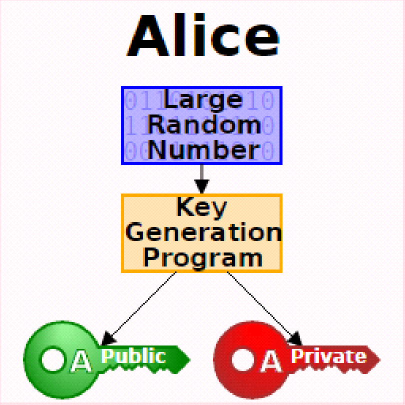
The one-wayness of ***F*** allow us to relate person with his/her **PrK** through the **PuK**. If ***F*** is 1-to-1, then the pair (**PrK**, **Puk**) is unique. So **PrK** could be reckoned as a unique secret parameter associated with certain person. This person can declare the possession or **PrK** by sharing his/her **PuK** as his public parameter related with **PrK** by and and at the same time not revealing **PrK**.

So, every user in asymmetric cryptography possesses key pair (**PrK**, **PuK**). Therefore, cryptosystems based on asymmetric cryptography are named as **Public Key CryptoSystems** (PKCS).

We will consider the same two traditional (canonical) actors in our study, namely Alice and Bob. Everybody is having the corresponding key pair ($\mathbf{PrK_A}$, $\mathbf{PuK_A}$) and ($\mathbf{PrK_B}$, $\mathbf{PuK_B}$) and are exchanging with their public keys using open communication channel as indicated in figure below.



Asymmetric Key Pair Generation



PrK and PuK are mathematically related by One-Way Function:

PuK = **F**(PrK)

F is one-way function - OWF if:

1.For given **F** defined by public parameters and **PrK** it is easy to compute **PuK** .

2.For given **F** and **PuK** it is infeasible to find **PrK** = $\mathbf{F}^{-1}$(**PuK**).

Public Parameters PP = (**p**, **g**): for **F** definition

PrK = *x* <-- randi ==> PuK = *a* = g^x mod p

2.Private Keys Prk and public Public Keys PuK generation.

PrK_A = x = randi(p-1)

a = g^x mod p

PuK_A = a = mod_exp(g,x,p)

PrK_B = y = randi(p-1)

b = g^y mod p

PuK_B = b = mod_exp(g,y,p)

1. Identification.

If person can prove that he/she knows **PrK** corresponding to his/her **PuK** without revealing any information about **PrK** then everybody can trust that he is communicating with person posessing (**PrK**, **Puk**) key pair. This kind of proof is named as ***Zero Knowledge Proof*** (ZKP) and plays a very important role in cryptography. It is very useful to realize identification, Digital Signatures and many other cryptographically secure protocols in internet. In many cryptographic protocols, especially in identification protocols **PrK** is named as **witness** and **PuK** as a **statement** for **PrK**.
Every actor is having the corresponding key pair (**PrK_A**, **PuK_A**) and (**PrK_B**, **PuK_B**) and are exchanging with their public keys using open communication channel as indicated in figure below.
Let Bob is sure that **PuK_A** is of Alice and wants to tell Alice that he intends to send her his photo with chamomile flowers dedicated for Alice. But he wants to be sure that he is communicating only with Alice itself and with nobody else. He hopes that at first Alice will prove him that she knows her secret **PrK_A** using ZKP protocol. In general, this protocol is named as identification protocol, is interactive and has 3 communications to exchange the following data named as ***commitment***, ***challenge*** and ***response***.

Registration phase: Bank generates PrK_A = **x** and PuK_A = **a** to Alice and hands over this data in smart card, or other crypto chip in Alice's smart phone, or in software for Smart ID.

Schnorr Id Scenario: **Alice** wants to prove **Bank** that she knows her Private Key - **PrK$_A$** which corresponds to her Public Key - **PuK$_A$** not revealing **PrK$_A$**: Zero Knowledge Proof - ZKP
Protocol execution between **Alice** and **Bank** has time limit.
Alice's computation resources has a limit --> protocol must be computationally effective.
PrK$_A$=x is called a **witness** and corresponding **PuK$_A$**=a=g^x **mod** p is called a **statement**.
This protocol is initiated by Alice and has the following three communications.

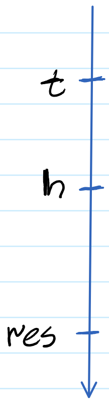
P(**x**, **a**) - Prover - **Alice** **V**(**a**) - Verifier - **Bank**

Public Parameters PP = (p, g)

1. **Alice** generates at random secret random number u=**randi(**p-2**)** and using **PP=(**p, g**)**
2. computes **commitment** t in the following way

$$t=g^u \textbf{ mod } p,\ t\in Z_{p-1}=\{0, 1, 2, \dots, p\text{-}2\}$$

Alice sends t *and* a to **Bank**.



Time

3. **Bank** generates at random **challenge** h=**randi(**p-2**)** and sends h to **Alice**.

4. **Alice** after receiving h computes her **response** res having her private key x together
5. with previously generated secret number u:

$$res=u+xh \textbf{ mod } p\text{-}1;\ res\in Z_{p-1}=\{0, 1, 2, 3, \dots, p\text{-}2\}$$

Alice sends r to the **Bank**.

After the third communication **Bank** verifies if the following identity holds:

$$g^{res}=ta^h \textbf{ mod } p.$$

P(**x**, **a**) - Prover - **Alice** **V**(**a**) - Verifier - **Bank**

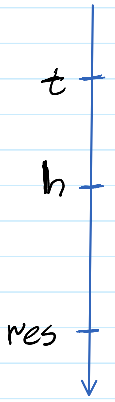
$x \leftarrow randi(\mathbb{Z}_{p-1})$; $a = g^x\ mod\ p$

$A:\ u \leftarrow randi(\mathbb{Z}_{p-1}) \quad \xrightarrow{t} \quad B:\ a$

$t = g^u\ mod\ p \quad \xleftarrow{h} \quad h \leftarrow rand(\mathbb{Z}_{p-1})$

$res = u + x \cdot h\ mod\ (p-1) \quad \xrightarrow{res} \quad$ Verifies if $g^{res} = t\, a^h\ mod\ p$

$$g^{res} = g^{(u+xh)\, mod\, (p-1)}\ mod\ p = g^u \cdot g^{xh}\ mod\ p =$$
$$= t \cdot a^h\ mod\ p.$$



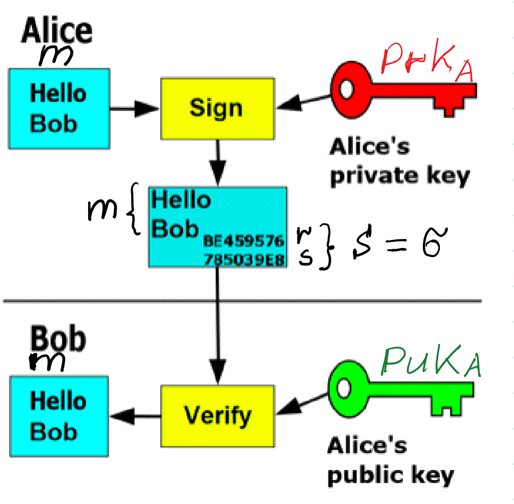
Time

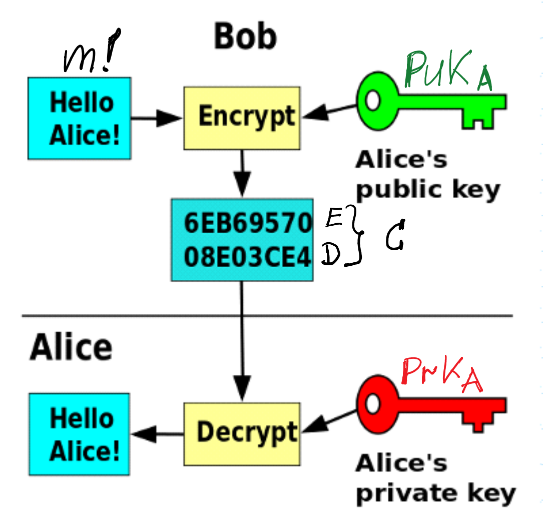
https://imimsociety.net/en/
https://imimsociety.net/en/14-cryptography

https://imimsociety.net/en/14-cryptography

Asymmetric Signing - Verification
S=Sign(PrK$_A$, m)
V=Ver(PuK$_A$, S, m), V$\in$\{True, False\}$\equiv$\{1, 0\}

Asymmetric Encryption - Decryption
c=Enc(PuK$_A$, m)
m=Dec(PrK$_A$, c)





In general, to create a signature on the message of any length ***M*** parties are using cryptographic secure H-function (message digest).
In Octave we use H-function

```
>> hd28('…')     % the input '…' of this function represents a string of symbols between the commas.
                 % the output of this function is decimal number having at most 28 bits.
```

Schnorr Signature Scheme (S-Sig).

Let ***M*** be a message in string format to be signed by **Alice** and sent to **Bank**: >> M='Hello Bob'
For signature creation **Alice** uses public parameters **PP=(*p*, *g*)** and
Alice's key pair is **PrK$_A$=*x***, **PuK$_A$=a**.

Alice chooses at random ***u***, 1<***u***<***p***-1 and computes first component ***r*** of his signature:

$$r=g^u \bmod p. \qquad (2.19)$$

Alice computes H-function value ***h*** and second component ***s*** of his signature:

$$h=\mathrm{H}(M||r), \qquad (2.20)$$

$$s=u+xh \bmod p\text{-}1. \qquad (2.21)$$

Alice's signature on ***h*** is **σ=(*r*,*s*)**. Then **Alice** sends ***M*** and **σ** to **Bank**.

After receiving ***M*** and **σ**, **Bank** according to (2.20) computes ***h'*** and verifies if

$$g^s \bmod p = ra^{h'} \bmod p. \qquad (2.22)$$

Symbolically this verification function we denote by

$$\mathrm{Ver}(a,\sigma,h)=\mathrm{V}\in\{True, False\}\equiv\{1, 0\}. \qquad (2.23)$$

This function yields ***True*** if (2.22) is valid <-- ***h*=*h'*** and **PuK$_A$=*F*(PrK$_A$)**.

```
>> m='Hello Bob'
```

```
m = Hello Bob
>> r=1234567
r = 1234567
>> cc=concat(m,r)
cc = Hello Bob1234567          % cc is a string type variable
>> cc=concat(m,'1234567')
cc = Hello Bob1234567
> h=hd28(cc)
h = 242519187
>> h=hd28('Hello Bob1234567')
h = 242519187
```

$$g^s = g^{(t+x\,h)\,mod(p-1)} mod\, p = g^t \cdot g^{x h'} mod\, p = r \cdot (g^x)^{h'} mod\, p =$$

$$= r \cdot a^{h'} mod\, p.$$

If $h' = h$ & $a = g^x mod\, p \Rightarrow$ equation(2.22) is valid $\Rightarrow$

$\Rightarrow$ signature is formed with $PrK = x$ corresponding to $PuK = a = g^x mod\, p.$

AKAP

A: $PrK = x$; $PuK = a$.

$PuK_B = b$; PuK_{TTP}.

$u \leftarrow randi(\mathbb{Z}_{p-1})$

$t_A = g^u \; mod \; p$

$t \leftarrow randi(\mathbb{Z}_{p-1})$

$r = g^t mod\, p$

$h = H(t_A \| r)$

$s = t + x\,h \; mod(p-1)$ $\xrightarrow{t_A,\ \sigma = (r, s)}$ B: $PrK_B = y$; $PuK_B = b$.

$Ver(a, \sigma, t_A) = True$

Till this place

① A browser verifies TTP signature on $\xleftarrow[cert_B]{t_B,\ \sigma_B = (R, S)}$ $\big(v \leftarrow randi(\mathbb{Z}_{p-1})$

$t \; = g^v mod\, p$

$t_B, \sigma_B = (R, S)$

TTP signature on PuK_B. $\leftarrow$ $Cert_B$

② A verifies B signature

$\sigma_B = (R, S)$ on t_B

③ A computes common secret key

$k_{AB} = (t_B)^u \bmod p$

$$k_{AB} = k = k_{BA}$$

$v \leftarrow randi(\mathbb{Z}_{p-1})$

$t_B = g^v \bmod p$

$\ell \leftarrow randi(\mathbb{Z}_{p-1})$

$R = g^\ell \bmod p$

$H = H(t_B || R)$

$S = \ell + yH \bmod (p-1)$

$Cert(PuK_B) = \underset{\uparrow \text{signed}}{Cert_B}$

Trusted Third Party – TTP

(PrK_{TTP}, PuK_{TTP})

Veri sign

$k_{BA} = (t_A)^v \bmod p$