Discrete Exponent Function - DEF (1/14)

The Discrete Exponent Function (**DEF**) used in cryptography firstly was introduced in the cyclic multiplicative group $Z_{p^*} = \{1, 2, 3, \ldots, p-1\}$, with binary multiplication operation *** mod** p, where p is prime number. Further the generalizations were made especially in *Elliptic Curve Groups* laying a foundation of *Elliptic Curve CryptoSystems* (ECCS) in general and in *Elliptic Curve Digital Signature Algorithm* (ECDSA) in particular.

Let g be a generator of Z_p^* then **DEF** is defined in the following way:

$$\mathbf{DEF}_g(x) = g^x \mathbf{\ mod\ } p = a;$$

DEF argument x is associated with the private key – **PrK** (or other secret parameters) and therefore we will label it in red and value a is associated with public key – **PuK** (or other secret parameters) and therefore we will label it in green.

In order to ensure the security of cryptographic protocols, a large prime number p is chosen. This prime number has a length of 2048 bits, which means it is represented in decimal as being on the order of 2^{2048}, or approximately $p \sim 2^{2048}$.

In our modeling with Octave, we will use p of length having only 28 bits for convenience. We will deal also with a strong prime numbers p.

T2. Fermat (little)Theorem. If p is prime, then [Sakalauskas, at al.]

$$z^{p-1} = 1 \bmod p$$

Discrete Exponent Function (2/14)

Definition. Binary operation *** mod** p in Z_p^* is an arithmetic multiplication of two integers called operands and taking the result as a residue by dividing by p.

For example, let $p = 11$, then $Z_p^* = \{1, 2, 3, \ldots, 10\}$, then 5 * 8 **mod** 11 = 40 **mod** 11 = 7, where $7 \in Z_p^*$.

In our example the residue of 40 by dividing by 11 is equal to 7, i.e., 40 = 3 * 11 + 7.
Then 40 **mod** 11 = (33 + 7) **mod** 11 = (33 **mod** 11 + 7 **mod** 11) **mod** 11 = (0 + 7) **mod** 11 = 7.
Notice that 33 **mod** 11 = 0 and 7 **mod** 11 = 7.

Definition: The integer g is a generator in Z_p^* if powering it by integer exponent values x all obtained numbers that are computed **mod** p generates all elements in in Z_p^*.

So, it is needed to have at least p-1 exponents x to generate all p-1 elements of Z_p^*. You will see that exactly p-1 exponents x is enough.

Discrete Exponent Function (3/14)

Let Γ be the set of generators in Z_p^*. How to find a generator in Z_p^*?

In general, it is a hard problem, but using strong prime p and *Lagrange theorem in group theory* the generator in Z_p^* can be found by random search satisfying two following conditions if p is strongprime.

For all $g \in \Gamma$

$$g^q \neq 1 \mathbf{\ mod\ } p; \text{ and } g^2 \neq 1 \mathbf{\ mod\ } p.$$

Fermat little theorem: If p is prime then for all integers i:

$$i^{p-1} = 1 \mathbf{\ mod\ } p.$$

Corollaries: 1. The exponent p-1 is equivalent to the exponent 0, since $i^0 = i^{p-1} = 1 \bmod p$.

2. Any exponent e can be reduced **mod** (p-1), i.e.

$$i^e \bmod p = i^{e \bmod (p-1)} \bmod p.$$

3. All non-equivalent exponents x are in the set $Z_{p-1} = \{0, 1, 2, \ldots, p-2\}$; +,-, *mod (p-1) and / mod(p-1) wth exception.

4. Sets Z_{p-1} and Z_p^* have the same number of elements.

Discrete Exponent Function (4/14)

In Z_{p-1} addition +, multiplication $*$ and subtraction - operations are realized **mod** (p-1).

Subtraction operation (h-d) **mod** (p-1) is replaced by the following addition operation (h + (-d)) **mod** (p-1)).

Therefore, it is needed to find -d **mod** (p-1) such that d + (-d) = 0 **mod** (p-1), then assume that

$$-d \bmod (p-1) = (p-1-d).$$

Indeed, according to the distributivity property of modular operation

$$(d + (-d)) \bmod (p-1) = (d + (p-1-d) \bmod (p-1) = (p-1) \bmod (p-1) = 0.$$

Then

$$(h-d) \bmod (p-1) = (h + (p-1-d)) \bmod (p-1)$$

```
>> p=11
p = 11
>> d=8;
>> md=p-1-d
md = 2
```

```
>> mod(d+md,p-1)
ans = 0
>> d+md
ans = 10
```

Discrete Exponent Function (5/14)

Statement: If greatest common divider between p-1 and i is equal to 1, i.e., gcd(p-1, i) = 1, then there exists unique inverse element i^{-1} **mod** (p-1) such that $i * i^{-1}$ **mod** (p-1) = 1. This element can be found by *Extended Euclide algorithm* or using *Fermat little theorem*. We do not fall into details how to find i^{-1} **mod** (p-1) since we will use the ready-made computer code instead in our modeling.

Division operation / **mod** (p-1) of any element in Z_{p-1} by some element i is replaced by multiplication $*$ operation with i^{-1} **mod** (p-1) if gcd(i, p-1) = 1 according to the *Statement* above.

To compute u/i **mod** (p-1) it is replaced by the following relation $u * i^{-1}$ **mod** (p - 1) since

$$u / i \bmod (p-1) = u * i^{-1} \bmod (p-1).$$

```
>> d=8;
>> d_m1=mulinv(d,p-1)
d_m1 = Inverse element does not exist
>> gcd(d,p-1)
ans = 2
>> d=9;
>> gcd(d,p-1)
ans = 1
>> d_m1=mulinv(d,p-1)
d_m1 = 9
>> mod(d*d_m1,p-1)
ans = 1
>> 1/9
ans = 0.1111
```

```
>> d=3;
>> d_m1=mulinv(d,p-1)
d_m1 = 7
>> mod(d*d_m1,p-1)
ans = 1
>> d*d_m1
ans = 21
>> mog(ans,p-1)
error: 'mog' undefined near line 1, column
1
>> mod(ans,p-1)
ans = 1
```

Discrete Exponent Function (6/14)

Example 1: Let for given integers u, x and h in Z_{p-1} we compute exponent s of generator g by the expression

$$s = u + xh.$$

Then

$$g^s \bmod p = g^{s \bmod (p-1)} \bmod p.$$

Therefore, s can be computed **mod** (p-1) in advance, to save a multiplication operations, i.e.

$$s = u + xh \bmod (p-1).$$

Example 2: Exponent s computation including subtraction by xr **mod** (p-1) and division by i in Z_{p-1} when gcd(i, p-1) = 1.

$$s = (h - xr)i^{-1} \bmod (p-1).$$

Firstly $d = xr$ **mod** (p-1) is computed:
Secondly $-d = -xr$ **mod** (p-1) = (p-1-d) is found.
Thirdly i^{-1} **mod** (p-1) is found.
And finally exponent $s = (h + (p-1-d))i^{-1}$ **mod** (p-1) is computed.

```
>> d=mod(x*r,p-1);
>> md=mod(-d,p-1);
>> i_m1=mulinv(i,p-1);
```

Discrete Exponent Function (7/14)

Referencing to Fermat little theorem and its corollaries, formulated above, the following theorem can be proved.

Theorem. If g is a generator in Z_p^* then **DEF** provides the following 1-to-1 mapping

$$\text{DEF}: Z_{p-1} \rightarrow Z_p^*.$$

Parameters p and g for **DEF** definition we name as Public Parameters and denote by **PP** = (p, g).

Example: Strong prime p = 11, p = 2 ∗ 5 + 1, then q = 5 and q is prime. Then p-1 = 10.

$$Z_{11}^* = \{1, 2, 3, \ldots, 10\}$$

$$Z_{10} = \{0, 1, 2, \ldots, 9\}$$

Discrete Exponent Function (8/14)

The results of any binary operation (multiplication, addition, etc.) defined in any finite group is named *Cayley table* including multiplication table, addition table etc.

Multiplication table of multiplicative group Z_{11}^* is represented below.

Multiplication tab. mod 11	Z_{11}^*									
*****	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Values of inverse elements in Z_{11}^*

1^{-1}= 1 mod 11
2^{-1}= 6 mod 11
3^{-1}= 4 mod 11
4^{-1}= 3 mod 11
5^{-1}= 9 mod 11
6^{-1}= 2 mod 11
7^{-1}= 8 mod 11
8^{-1}= 7 mod 11
9^{-1}= 5 mod 11
10^{-1}= 10 mod 11

Discrete Exponent Function (9/14)

The table of exponent values for p = 11 in Z_{11}^* computed **mod** 11 and is presented in table below.
Notice that according to Fermat little theorem for all $z \in Z_{11}^*$, $z^{p-1} = z^{10} = z^0 = 1$ **mod** 11.

Exponent tab. mod 11	Z_{11}^*											
^	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	*10*	
1	1	1	1	1	1	1	1	1	1	1	1	
2	1	2	4	8	5	10	9	7	3	6	1	$2^2 \neq 1$ **mod** 11 & $2^5 \neq 1$ **mod** 11
3	1	3	9	5	4	1	3	9	5	4	1	
4	1	4	5	9	3	1	4	5	9	3	1	
5	1	5	3	4	9	1	5	3	4	9	1	
6	1	6	3	7	9	10	5	8	4	2	1	$6^2 \neq 1$ **mod** 11 & $6^5 \neq 1$ **mod** 11
7	1	7	5	2	3	10	4	6	9	8	1	$7^2 \neq 1$ **mod** 11 & $7^5 \neq 1$ **mod** 11
8	1	8	9	6	4	10	3	2	5	7	1	$8^2 \neq 1$ **mod** 11 & $8^5 \neq 1$ **mod** 11
9	1	9	4	3	5	1	9	4	3	5	1	
10	1	10	1	10	1	10	1	10	1	10	1	

Discrete Exponent Function (10/14)

Notice that there are elements satisfying the following different relations, for example:

$$3^5 = 1 \textbf{ mod } 11 \text{ and } 3^2 \neq 1 \textbf{ mod } 11.$$

The set of such elements forms a subgroup of prime order $q = 5$ if we add to these elements the *neutral group element* 1.

This subgroup has a great importance in cryptography we denote by

$$G_5 = \{1, 3, 4, 5, 9\}.$$

The multiplication table of G_5 elements extracted from multiplication table of Z_{11}^* is presented below.

Multiplication tab. mod 11	*G5*				
*	**1**	**3**	**4**	**5**	**9**
1	1	3	4	5	9
3	3	9	1	4	5
4	4	1	5	9	3
5	5	4	9	3	1
9	9	5	3	1	4

Values of inverse elements in G_5

$1^{-1} = 1$ mod 11
$3^{-1} = 4$ mod 11
$4^{-1} = 3$ mod 11
$5^{-1} = 9$ mod 11
$9^{-1} = 5$ mod 11

Exponent tab. mod 11	*G5*					
^	**0**	**1**	**2**	**3**	**4**	**5**
1	1	1	1	1	1	1
3	1	3	9	5	4	1
4	1	4	5	9	3	1
5	1	5	3	4	9	1
9	1	9	4	3	5	1

Discrete Exponent Function (11/14)

Notice that since G_5 is a subgroup of Z_{11}^* the multiplication operations in it are performed **mod** 11.
The exponent table shows that all elements {3, 4, 5, 9} are the generators in G_5.
Notice also that for all $\gamma \in \{3, 4, 5, 9\}$ their exponents 0 and 5 yields the same result, i.e.

$$\gamma^0 = \gamma^5 = 1 \text{ mod } 11.$$

This means that exponents of generators γ are computed **mod** 5.

This property makes the usage of modular groups of prime order q valuable in cryptography since they provide a higher-level security based on the stronger assumptions we will mention later.

Therefore, in many cases instead the group Z_p^* defined by the prime (not necessarily strong prime) number p the subgroup of prime order G_q in Z_p^* is used.

In this case if p is strong prime, then generator γ in G_q can be found by random search satisfying the following conditions

$$\gamma^q = 1 \textbf{ mod } p \text{ and } \gamma^2 \neq 1 \textbf{ mod } p.$$

Analogously in this generalized case this means that exponents of generators γ are computed **mod** q. In our modeling we will use group Z_p^* instead of G_q for simplicity.

Discrete Exponent Function (12/14)

Let as above p=11 and is strong prime and generator we choose g = 7 from the set Γ={2, 6, 7, 8}.
Public Parameters are **PP**=(**11**,**7**), Then $\mathbf{DEF}_g(x)$ = $\mathbf{DEF}_7(x)$ is defined in the following way:

$$\mathbf{DEF}_7(x) = 7^x \ \mathbf{mod}\ 11 = a;$$

$\mathbf{DEF}_7(x)$ provides the following 1-to-1 mapping, displayed in the table below.

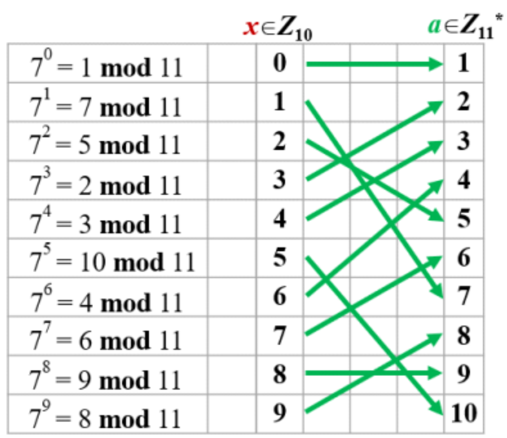
x	0	1	2	3	4	5	6	7	8	9	*10*	*11*	*12*	*13*	*14*
7^x **mod** p = a	1	7	5	2	3	10	4	6	9	8	*1*	*7*	*5*	*2*	*3*

You can see that a values are repeating when x = 10, 11, 12, 13, 14, etc. since exponents are reduced **mod** 10 due to *Fermat little theorem*.

The illustration why 7^x **mod** p values are repeating when x = 10, 11, 12, 13, 14, etc. is presented in computations below:
10 mod 10 = 0; 710 = 70 = 1 mod 11 = 1.
11 mod 10 = 1; 711 = 71 = 7 mod 11 = 7.
12 mod 10 = 2; 712 = 72 = 49 mod 11 = 5.
13 mod 10 = 3; 713 = 73 = 343 mod 11 = 2.
14 mod 10 = 4; 714 = 74 = 2401 mod 11 = 3.
etc.

Discrete Exponent Function (13/14)

For illustration of 1-to-1 mapping of $\mathbf{DEF}_7(x)$ we perform the following step-by-step computations.



It is seen that one value of x is mapped to one value of a.

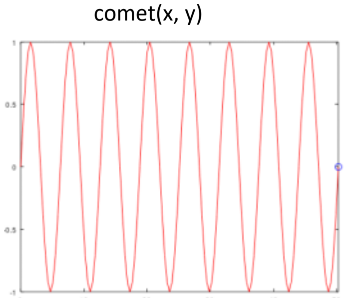
Discrete Exponent Function (14/14)

But the most in interesting think is that **DEF** is behaving like a *pseudorandom function*.

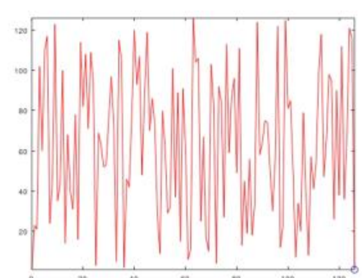
It is a main reason why this function is used in cryptography - classical cryptography.

To better understand the pseudorandom behaviour of **DEF** we compare the graph of "regular" **sine** function with "pseudorandom" **DEF** using Octave software.

```
>> p128sin
xrange = 16 * pi;
step = xrange/128;
x = 0:step:xrange;
y = sin(x);
comet(x, y)
```



```
>> p128def
p = 127;
g = 23;
x = 0:p-1;
a = mod_expv(g, x, p);
comet(x, a)
```



Example 1: Let for given integers u, x and h in Z_{p-1} we compute exponent s of generator g by the expression

$$s = u + xh.$$

Then

$$g^s \bmod p = g^{s \bmod (p-1)} \bmod p.$$

Therefore, s can be computed **mod** (p-1) in advance, to save a multiplication operations, i.e.

$$s = u + xh \bmod (p-1).$$

```
>> p=genstrongprime(28)
p = 242502683
>> q=(p-1)/2
q = 121251341
>> g=2
g = 2
>> mod_exp(g,q,p)
ans = 242502682
>> x=randi(p-1)
x = 4.8906e+07
>> x=int64(randi(p-1))
x = 165532552
```

```
>> u=int64(randi(p-1))
u = 74661797
>> h=int64(randi(p-1))
h = 194373549
>> xh=mod(x*h,p-1)
xh = 218184446
>> upxh=mod(u+xh,p-1)
upxh = 50343561
>> s=upxh
s = 50343561
```

```
>> snr=int64(u+x*h)
snr = 32175149681928845
>> mod(snr,p-1)
ans = 50343561
```

Example 2: Exponent s computation including subtraction by xr **mod** (p-1) and division by i in Z_{p-1} when gcd(i, p-1) = 1.

$$s = (h - xr)i^{-1} \bmod (p-1).$$

Firstly $d = xr$ **mod** (p-1) is computed:
Secondly $-d = -xr$ **mod** (p-1) = (p-1-d) is found.
Thirdly i^{-1} **mod** (p-1) is found.
And finally exponent $s = (h + (p-1-d))i^{-1}$ **mod** (p-1) is computed.

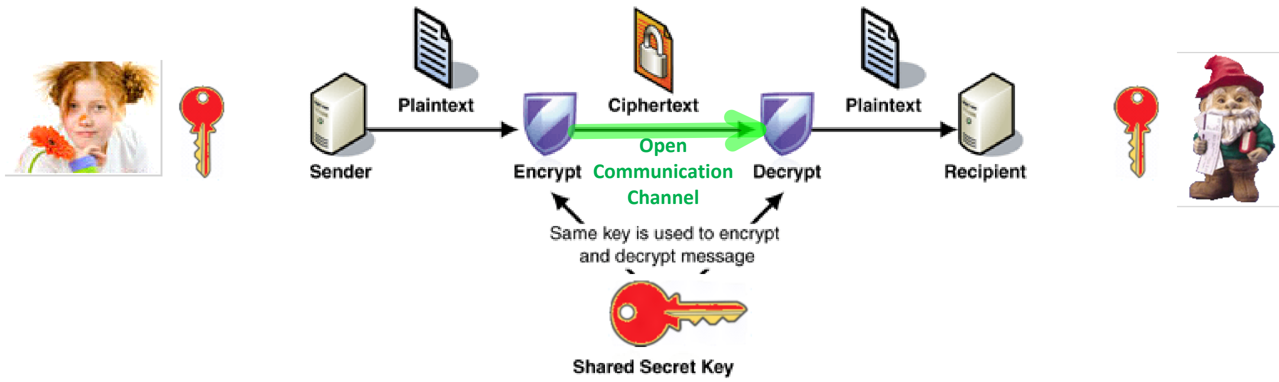
```
>> r=int64(randi(p-1))
r = 212560238
>> i=int64(randi(p-1))
i = 64538497
>> xr=mod(x*r,p-1)
xr = 98263592
>> mxr=mod(-xr,p-1)
mxr = 144239090
>> xrpmr=mod(xr+mxr,p-1)
xrpmr = 0
>> hpmxr=mod(h+mxr,p-1)
hpmxr = 96109957
```

```
>> i_m1=mulinv(i,p-1)
i_m1 = 196196855
>> gcd(i,i_m1)
ans = 1
>> mod(i*i_m1,p-1)
ans = 1
>> s=mod(hpmxr*i_m1,p-1)
s = 131208547
```

$$s = (h - xr)i^{-1} \bmod (p-1).$$

$g^s \bmod p = \ldots$

Firstly $d = xr$ **mod** (p-1) is computed:
Secondly $-d = -xr$ **mod** (p-1) = (p-1-d) is found.
Thirdly i^{-1} **mod** (p-1) is found.
And finally exponent $s = (h + (p-1-d))i^{-1}$ **mod** (p-1) is computed.



Diffie-Hellman Key Agreement Protocol (DH KAP)

Public Parameters PP=(*p*,*g*)

$u \leftarrow rand(Z_p^*)$

$g^u \bmod p = t_A$ → t_A

$v \leftarrow rand(Z_p^*)$

t_B ← $t_B = g^v \bmod p$

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

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

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

Security considerations: if someone can compute for example a secret param. u generated by A the he/she can compute secret key k by intercepting t_B

Adv.: $(t_B)^u \bmod p = k$.

If p is generated large enough, e.g. $p \sim 2^{2048} \approx 10^{700}$, $|p| = 2048$ bits the to find u when p, g and t_A are given is infeasible with classical computers.

It is ifeasible to compute u from the equation $g^u \bmod p = t_A$ by having p, g and t_A.

The problem to find u when p, g and t_A are given is called a discret logarithm problem — DLP

$$dlog_g (g)^u \bmod p = u \cdot dlog_g (g) \bmod p = u \cdot 1 \bmod p = u.$$