

5 D U N 6 6 H E D X C 4 I Q 0 M
D 0 V 8 Y F E B K V U O K X F R
F 5 A N J 0 9 G A 4 G B 0 B Q
W C R R H T R 4 N 0 O V V 7
X C O O 5 A P Z O W B S G S
E 6 V R S **CRYPTOLOGY** E
N I 8 2 C Q N B A 7 B L 0 X
T 9 U E L P S G 5 Y U U A 8 L 5
Q I S V **LECTURE NOTES** 8
N 6 M 4 T Y Y 1 B 9 6 N 3 V
Z E V L 8 I B U 5 2 Z A 6 R
8 1 **FRANK SCHNEIDER** K
F 8 5 8 Q S R G K D 8 P C E Y
3 7 Z E X Q D 3 0 N 5 K Z 3 8
S 1 E J K M C Q X X C 6 N F

Copyright © 2015 Frank Schneider

This file was created as an unofficial transcript of the lecture “Cryptology” by TANJA LANGE at the TECHNISCHE UNIVERSITEIT EINDHOVEN TU/e.

These lecture notes are only meant for personal use.

The figures used in this file are self-provided (adapted from the original lecture notes by TANJA LANGE and RUBEN NIEDERHAGEN) if not attributed otherwise.

September 26, 2015

Contents

1	Introduction	5
1.1	Cryptography vs. Cryptoanalysis	5
1.2	Caesar Cipher	5
1.3	Two types of crypto systems	6
2	Mathematic Repetition	9
2.1	Modulus Calculation	9
2.1.1	Multiplication	9
2.1.2	Exponentiation	9
2.1.3	Inverses	10
2.2	Lagrange's Theorem	11
3	RSA	13
3.1	Generating the Keys	13
3.2	Encryption and Decryption of message	14
3.2.1	Encryption	14
3.2.2	Decryption	14
3.3	Making the Algorithm Faster	15
4	Finite Fields	17

5	Diffie-Hellman Key Exchange	25
5.1	Introduction	25
5.1.1	Computational DH Problem (CDHP)	26
5.1.2	Decisional DH Problem (DDHP)	26
5.1.3	Discrete Logarithm Problem (DLP)	26
5.2	ElGamal Encryption	27
5.2.1	ElGamal Signature Scheme	27
5.3	Pohlig-Hellman Attack	28
5.3.1	Pohlig-Hellman & DDH	29
5.4	Index Calculus	31
5.4.1	Logjam	33
	Index	35

1. Introduction

1.1 Cryptography vs. Cryptoanalysis

Cryptography: The constructive part (e.g. building systems, “constructing the encryption”).


Cryptanalysis: The destructive part (e.g. breaking systems, “cracking the code”).

This course covers both parts of Cryptology. Both parts are needed in deployed systems, since we want to know how efficient the system is **and** how hard it is to break it:

■ Example 1.1

- 2^{60} is inconvenient to compute, but totally doable;
- 2^{70} is doable for academic clusters;
- 2^{80} is doable for the NSA;
- 2^{128} nobody can do that (including a security margin);

■

 These numbers depend on the model of computation. Algorithms for example have different runtimes on quantum computers.

1.2 Caesar Cipher

A standard example in cryptology is a CAESAR CIPHER. In this example ALICE (denoted A in figure 1.1) wants to transmit a secret message ($E(m)$) to BOB (B), since the eavesdropper EVE (E) can see or hear everything on the channel.

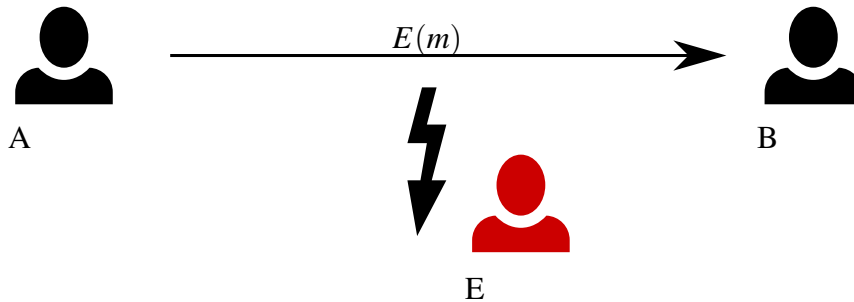


Figure 1.1: Encrypted communication

So ALICE and BOB decide to use a simple form of encryption - the CAESARS CIPHER. They replace the letters of the message by a letter some fixed number (in this case it is 3) of positions down the alphabet (see figure 1.2).



Figure 1.2: Idea of CAESARS CIPHER

■ Example 1.2

The (original) message

HELLO (Plaintext), would be encrypted into

KHOOR (Ciphertext) ■

This method of encryption is easy to cryptanalyze.

It is possible to turn this into a keyed encryption: $E_k(m)$, where the key k gives the shifting distance. The decryption therefore is: $D_k(c)$, which should satisfy

$$D_k(E_k(m)) = m.$$

1.3 Two types of crypto systems

There are two types of crypto systems: *symmetric* and *asymmetric* (= public key) systems. In the *symmetric* crypto A and B share the same key (e.g. the shifting distance of the CAESARS CIPHER, 128 bits in AES).

In *public key* crypto each user has 2 keys:

- A *public* one which can be used by anybody to encrypt messages to this user and
- A *private* (= secret key) one which the user uses to decrypt messages.

Therefore in *asymmetric* systems this means $c = E_{P_k}(m)$, $D_{S_k}(c) = m$, with P_k the *public* key and S_k the *secret* key. P_k and S_k can be of very different nature.

In a good crypto system one should not be able to recover S_k from P_k (= *complete break*) or to decrypt c without knowing S_k .

R This means that there are two possibilities of cryptanalyzing a system. One can either find the *secret* key or find the message. Obviously the first case means a *complete break*, meaning that every other message can be decrypted as well.

We will see different crypto systems such as RSA, ELLIPTIC-CURVE CRYPTOGRAPHY (ECC) and McELIECE ENCRYPTION including attacks on these.

Crypto is also about authentication, e.g. digital signatures.

2. Mathematic Repetition

2.1 Modulus Calculation

Notation 2.1.

\mathbb{Z} : integers $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$

\mathbb{Z}/n : residue classes of \mathbb{Z} modulo n

$\mathbb{Z}/6 = \{0, 1, 2, 3, 4, 5\} = \{-2, -1, 0, 1, 2, 3\}$.

We use integers to represent classes, so 0 stands for the set of integers which are congruent to 0 modulo 6, i.e. $\{0, 6, 12, 18, \dots, -6, -12, \dots\}$. You might have learned this as $\bar{0}$ to denote the class.

$7 \equiv 1 \pmod{6}$ ($\hat{=}$ 7 is congruent to 1 modulo 6). Usually we want the small representative on the right side; depends on set chosen, e.g. $17 \equiv 5 \equiv -1 \pmod{6}$.

2.1.1 Multiplication

If we have a multiplication with modulus n , e.g. $a \cdot b \pmod{n}$. We can compute the result and then reduce - or we could reduce at any intermediate step. $17 \cdot 35 \equiv -1 \cdot (-1) \equiv 1 \pmod{6}$.

2.1.2 Exponentiation

If we want to compute an exponentiation with modulus n : $a^b \pmod{n}$, we write b in binary:

$$b = \sum_{i=0}^{l-1} b_i 2^i; b_i \in \{0, 1\} \text{ with } b_{l-1} = 1$$

■ Example 2.2

$$17 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0;$$

$$35 = 1 \cdot 2^5 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

■

$$a^b = a^{\sum b_i \cdot 2^i} = \left(\dots \left((a^{b_{l-1} \cdot 2 + b_{l-2}})^{2 + b_{l-3}} \right)^{2 + \dots + 2 + b_0} \right)$$

Square and Multiply

```

 $c \leftarrow a$  ;
for  $i = l - 2$  down to 0 do
     $c \leftarrow c^2 \bmod n$  ;
    if  $b_i = 1$  then
         $c \leftarrow c \cdot a \bmod n$  ;
    end
    output  $c$  ;
end

```

Algorithm 1: Square and Multiply - pseudocode

R This algorithm reduces at every intermediate step, making the exponentiation a lot simpler and faster.

2.1.3 Inverses

Sometimes we need to calculate inverses, e.g. divide equation mod 7 by 4, this means undo a multiplication by 4. What we are looking for is to find an integer $a \bmod 7$, such that $4 \cdot a \equiv 1 \bmod 7$, other notation $a^{-1} \equiv 4 \bmod 7$.

Here $a \equiv 2 \bmod 7$ because $4 \cdot 2 \equiv 8 \equiv 1 \bmod 7$.

In general, use EUCLIDEAN ALGORITHM to compute a .

R Look up XGCD-EXTENDED EUCLIDEAN ALGORITHM.

Inverses do not exist if the numbers are not coprime, e.g. 4 is not inversible mod 6.

Notation 2.3. Set of invertible integers mod n is denoted by $(\mathbb{Z}/n)^*$

■ Example 2.4

- $(\mathbb{Z}/7)^* = \{1, 2, 3, 4, 5, 6\}$ with inverses $1^{-1} \equiv 1 \bmod 7$, $2^{-1} \equiv 4 \bmod 7$, $3^{-1} \equiv 5 \bmod 7$, $6^{-1} \equiv 6 \bmod 7$; these are exactly the integers coprime with 7.
- $(\mathbb{Z}/6)^* = \{1, 5\}$
- $(\mathbb{Z}/p)^* = \{1, 2, 3, \dots, p-1\}$ for p prime.

■

EULER'S PHI FUNCTION gives the size of $|(\mathbb{Z}/n)^*| = \varphi(n)$.

■ Example 2.5

- $\varphi(7) = 6$
- $\varphi(6) = 2$
- $\varphi(p) = p - 1$

■

For the multiplication of two primes p, q ($p \neq q$) you can calculate the PHI FUNCTION as follows:

$$\begin{array}{ccccccccc}
 0 & 1 & 2 & 3 & 4 & \dots & q-1 & & \\
 q & q+1 & q+2 & q+3 & \dots & & 2q-1 & & \\
 2q & \dots & & & & & & & \\
 \vdots & & & & & & & & \\
 (p-1)q & & & & & & p \cdot q - 1 & &
 \end{array}$$

The numbers in the first column are all divisible by q (there are p of them). They can be removed (since p and q are primes, this means that only numbers which are multiples of p or q are not coprime).

Do the same with p :

$$0 \quad 1 \quad 2 \quad \dots \quad p-1$$

this removes q values,

$$\phi(p \cdot q) = p \cdot q - p - q + 1$$

Since the 0 is removed twice, we have to add +1.

■ Example 2.6

- $\phi(2 \cdot 3) = 2 \cdot 3 - 2 - 3 + 1 = 2$ (same)
- $\phi(p^2) = p^2 - p$, in general:
- $\phi(p^b) = p^b - p^{b-1}$
- if you have: $n = \prod_{i=0}^{l-1} p_i^{e_i}$, $p_i \neq p_j$ then ($e_i \in \mathbb{N} \leq 1$)

$$\phi(n) = \prod (p_i^{e_i} - p_i^{e_i-1})$$

Test: $\phi(p \cdot q) = (p-1)(q-1) = p \cdot q - p - q + 1$ as above.

■

2.2 Lagrange's Theorem

Theorem 2.7 — Lagrange's Theorem. Let G be a finite group of size $|G| = l$ then for any $a \in G$ we have

$$a^l = 1,$$

where 1 is the neutral element and G is written multiplicatively

■ **Example 2.8** $a^6 \equiv 1 \pmod{7}$ for $a \in (\mathbb{Z}/7)^*$ ■

3. RSA

The name RSA comes from the initials of RIVEST, SHAMIR, ADEMAN. In 1977 it was the first encryption and signature system with a public key.

3.1 Generating the Keys

We pick two large primes $p \neq q$, put $n = p \cdot q$ and compute $\varphi(n) = (p-1)(q-1)$.

Now pick an integer e , which is coprime with $\varphi(n)$.

Compute $e^{-1} \equiv d \pmod{\varphi(n)}$ using XGCD.

RSA public key: (e, n) private key: d

■ Example 3.1

$$p = 5, q = 7,$$

$$n = 5 \cdot 7 = 35$$

$$\varphi(n) = (5-1)(7-1) = 24$$

- Pick $e = 5$, $\gcd(5, 24) = 1$,
 $5^{-1} \equiv 5 \pmod{24}$, so $d = 5$
- Other options, e.g. $e = 7$. Compute \gcd :

$$\begin{array}{rrrr} 24 & 1 & 0 & \\ 7 & 0 & 1 & q=3 \\ 3 & 1 & -3 & q=2 \\ 1 & -2 & 7 & \\ r\uparrow & a\uparrow & b\uparrow & \end{array}$$

For every row, q is the number of times the first item in the row “fits into” the item above it. So for 24 and $e = 7$ this means: $q = 3$ (since $7 \cdot 3 = 21$ with rest 3). You

then subtract 3 times this row from the previous row, or more general: subtract q times row from previous.

Every row now satisfies that $r = a \cdot 24 + b \cdot 7$, e.g. $1 = -2 \cdot 24 + 7 \cdot 7 \Rightarrow 7^{-1} \equiv 7 \pmod{24}$

Then $(5, 35)$ public key and $d = 5$ secret key or $(7, 35)$ public key and $d = 7$ secret key.

R It is pure coincidence (and due to the fact that we use small numbers) that d and e are equivalent)

■

3.2 Encryption and Decryption of message

3.2.1 Encryption

We want to encrypt the message $m < n, m \in \mathbb{N}$.

$$c \equiv m^e \pmod{n}$$

R At this point in an implementation we would use the Square and Multiply method.

3.2.2 Decryption

To decrypt of c , we calculate:

$$m' \equiv c^d \pmod{n}$$

R We now want to proof that the decryption undos the encryption, or as we stated earlier $D(E(m)) = m$, or in this case $m' = m$

This decryption works because

$$m' \equiv c^d \equiv (m^e)^d \equiv m^{e \cdot d} \equiv m^{1+k\varphi(n)} \equiv m \pmod{n}$$

where $e \cdot d = 1 \pmod{\varphi(n)}$, i.e. $e \cdot d = 1 + k \cdot \varphi(n)$ for some k .

For the last step, we used LAGRANGE with group $(\mathbb{Z}/n)^*$ because there $m^{\varphi(n)} \equiv 1 \pmod{n}$. You can check that $m^{1+k\varphi(n)} \equiv m \pmod{n}$ even when $\gcd(m, n) \neq 1$.

To show this, use the CHINESE REMAINDER THEOREM (CRT)

$$m \equiv 0 \pmod{p} \quad m \equiv a \pmod{q}, a \neq 0, \text{ then } m^{e \cdot d} \equiv 0^{e \cdot d} \equiv 0 \pmod{p}$$

In $(\mathbb{Z}/q)^*$ we have $a^{q-1} \equiv 1 \pmod{q}$.

Thus

$$m^{e \cdot d} \equiv a^{e \cdot d} \equiv a^{1+k\varphi(n)} \equiv a^{1+(p-1)(q-1)} \equiv a \cdot \underbrace{(a^{q-1})^{p-1}}_{=1} \equiv a \cdot 1^{p-1} \equiv a \pmod{q}$$

$$m^{e \cdot d} \equiv 0 \equiv m \pmod{p}$$

$$m^{e \cdot d} \equiv a \equiv m \pmod{q},$$

thus CRT gives $m^{e \cdot d} \equiv m \pmod{n}$.

Last case $0^{e \cdot d} \equiv 0 \pmod{d} \Rightarrow$ RSA gives $m' = m$.

3.3 Making the Algorithm Faster

To get a faster encryption, we can use a small e when generating the key.

Choosing small d gives EVE your key; but we can decrypt faster using RSA-CRT: Compute

$$c^d \equiv m_p \pmod{p} \qquad c^d \equiv m_q \pmod{q}$$

This is faster because the operands are smaller (naively this save a factor of 4 in each computation, do this twice, so save factor of 2).

Combine m_p and m_q using CRT to get m .

We can save more effort by noticing that $c^{p-1} \equiv 1 \pmod{p}$, so compute $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$; d_p and d_q have half the bit length of d , so the SQUARE AND MULTIPLY loop is half as long in computing

$$c^{d_p} \equiv m_p \pmod{p} \qquad c^{d_q} \equiv m_q \pmod{q}$$

Combine m_p and m_q to m using CRT. This needs just one multiplication given $u \equiv p^{-1} \pmod{q}$ as precomputed value.

Then $m = m_p q \cdot v + p \cdot u \cdot m_q$ with $v \equiv q^{-1} \pmod{p}$. “Naively” because with KARATSUBA, doublelength costs only 3 times as much and for very long integers the FAST FOURIER TRANSFORMATION (FFT) takes only twice as long.

4. Finite Fields

Definition 4.1 — Fields. A set K is a *field* with respect to \circ and \diamond , denoted (K, \circ, \diamond) , if

i (K, \circ) is an abelian group:

- **closure** for all $a, b \in K \Rightarrow a \circ b \in K$
- **associativity** $(a \circ b) \circ c = a \circ (b \circ c)$ $a, b, c \in K$
- **identity** there is a $e_0 \in K$ such that for all $a \in K$, $a \circ e_0 = e_0 \circ a = a$
- **inverse** for all $a \in K$, there is a $b \in K$, such that $a \circ b = e_0$
- **commutative** $a \circ b = b \circ a$

ii $(K^* = K \setminus \{e_0\}, \diamond)$ is an abelian group

iii the distributive law holds in K :

$$a \diamond (b \circ c) = a \diamond b \circ a \diamond c$$

■ Example 4.2

- $(\mathbb{N}, +, \cdot)$ is **NOT** a finite field (e.g. there is no inverse for $+$)
- $(\mathbb{Z}, +, \cdot)$ is **NOT** a finite field (e.g. there is no inverse for \cdot)
- $(\mathbb{Q}, +, \cdot)$ is a finite field
- $(\mathbb{R}, +, \cdot)$ is a finite field
- $(\mathbb{C}, +, \cdot)$ is a finite field
- $K = \{0, 1\}$ is the smallest set we can get:

$+$	0	1
0	0	1
1	1	0

\cdot	0	1
0	0	0
1	0	1

relating to:

\circ	e_\circ	e_\diamond
e_\circ	e_\circ	e_\diamond
e_\diamond	e_\diamond	e_\diamond

\diamond	e_\circ	e_\diamond
e_\circ	e_\circ	e_\diamond
e_\diamond	e_\diamond	e_\diamond

The $+$ corresponds to XOR, the \cdot to AND

■

Definition 4.3 — Subfield. If (K, \circ, \diamond) and (L, \circ, \diamond) are fields and $K \subseteq L$ then K is a *subfield* of L .

R \Rightarrow We can add elements of L to and multiply them with elements of $K \Rightarrow L$ is a vectorspace over K (other properties work because of the distributive laws).

Definition 4.4 — Extension Degree. Let L be a field and let K be a subfield of L . The *extension degree* $[L : K]$ is defined as $\dim_K L$, the dimension of L as a K vectorspace.

Definition 4.5 — Characteristic. Let K be a field. The *characteristic* of K , denoted $\text{char}(K)$, is the smallest positive integer m such that $\underbrace{e_\diamond \circ e_\diamond \circ \dots \circ e_\diamond}_{m \text{ copies of } e_\diamond \text{ denoted as } [m]e_\diamond} = e_\diamond$; if no such integer exists, $\text{char}(K) = 0$.

Lemma 4.6 The characteristic of a field is 0 or a prime

Proof. Let $\text{char}(K) = n = a \cdot b$ $1 < a, b < n$. Then

$$e_\diamond = [m]e_\diamond = [a \cdot b]e_\diamond = [a]e_\diamond \diamond [b]e_\diamond$$

Since a field has no zero divisors it must be that $[a]e_\diamond = e_\diamond$ or $[b]e_\diamond = e_\diamond$ \nmid to minimality of $\text{char}(K) = n$. \blacksquare

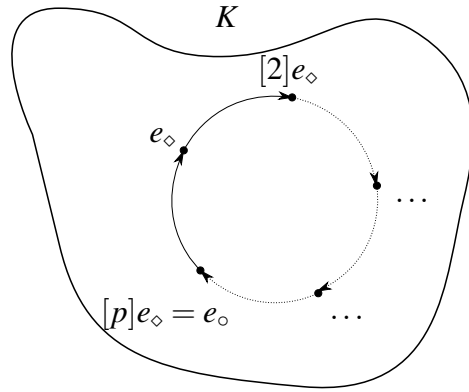
R In the proof of Lemma 4.6 we repeatedly used the distributive law:

$$\begin{aligned} & [a]e_\diamond \diamond [b]e_\diamond \\ \iff & (e_\diamond \circ [a-1]e_\diamond) \diamond [b]e_\diamond \iff \underbrace{e_\diamond \circ [b]e_\diamond}_{= [b]e_\diamond, \text{ since } e_\diamond \text{ is the neutral element for } \diamond} \circ [a-1]e_\diamond \diamond [b]e_\diamond \\ \iff & [b]e_\diamond \circ (e_\diamond \circ [a-2]e_\diamond) \diamond [b]e_\diamond \iff \dots \iff \underbrace{[b]e_\diamond \circ [b]e_\diamond \circ \dots \circ [b]e_\diamond}_{a \text{ times} \rightarrow [a \cdot b]e_\diamond} \end{aligned}$$

Lemma 4.7 A finite field K has characteristic p for some prime p

Proof. Since K is finite, there must be $i, j \in \mathbb{N}$ with $[i]e_\diamond = [j]e_\diamond$. Let $i > j$ then $[i-j]e_\diamond = e_\diamond$ and so $\text{char}(K) \mid (i-j)$. \blacksquare

Let K be a finite field. We will now explore its structure. We know already: $\text{char}(K) = p$ for a prime p , and there exists $e_\diamond, e_\diamond \in K$ with $e_\diamond \neq e_\diamond$. Since K is closed under \circ we do also find $[2]e_\diamond, [3]e_\diamond, \dots, [p-1]e_\diamond, [p]e_\diamond = e_\diamond, [p+1]e_\diamond = e_\diamond, \dots$ a cyclic subgroup of order p of (K, \circ) . Multiplying two such elements $[i]e_\diamond \diamond [j]e_\diamond = [ij]e_\diamond$ again gives us an element of the set $\{[i]e_\diamond \mid 0 \leq i < p\}$. The scalars are considered modulo p because $[p]e_\diamond = e_\diamond$. Since p is a prime, $i \cdot j \not\equiv 0 \pmod p$ for $0 < i, j < p$. This means that $\{[i]e_\diamond \mid 0 < i < p\}$ forms a subgroup of K^* (the multiplicative group in K ; $K^* = K \setminus \{e_\diamond\}$).



If two structures (groups, rings, fields,...) behave exactly the same way so that one can give a one-to-one map between them, mathematicians call these two structures *isomorphic*. Our considerations have found a subfield of K which is isomorphic to $\mathbb{Z}/p\mathbb{Z}$ with map $[i]e_\diamond \mapsto i + p\mathbb{Z}$.

Definition 4.8 — Prime Field. Let K be a field. The smallest subfield contained in K is called the *prime field* of K .

Lemma 4.9 Let K be a finite field of $\text{char}(K) = p$.

The prime field of K is isomorphic to $\mathbb{Z}/p\mathbb{Z}$

$$\begin{array}{ll} e_\circ \mapsto 0 & \text{in } \mathbb{Z}/p\mathbb{Z} \\ e_\diamond \mapsto 1 & \text{in } \mathbb{Z}/p\mathbb{Z} \end{array}$$

Above we found that an extension field can be considered as a vectorspace over its subfield. From now on we identify the prime field of a finite field with $\mathbb{Z}/p\mathbb{Z}$ and write 0 for e_\circ and 1 for e_\diamond . Let $[K : \mathbb{Z}/p\mathbb{Z}] = n$, i.e., the dimension of K as a vectorspace over $\mathbb{Z}/p\mathbb{Z}$ is n . This means that there exists a basis of n linearly independent “vectors” $\alpha_1, \alpha_2, \dots, \alpha_n$. This being a basis means that every element in K can be written in a unique way as $\sum_{i=1}^n c_i \alpha_i$ with $c_i \in \mathbb{Z}/p\mathbb{Z}$. The p^n different choices for $(c_1, c_2, \dots, c_n) \in \mathbb{Z}/p\mathbb{Z}^n$ mean that K has p^n elements.

Lemma 4.10 Let K be a finite field. There exists a prime p and an integer $n \in \mathbb{N}_{>0}$ such that $|K| = p^n$ and $\text{char}(K) = p$.

Notation 4.11. A field of characteristic p and dimension n is \mathbb{F}_{p^n} or $GF(p^n)$ (for “GALOIS field”)

This implies that every finite field has a prime power as its cardinality, so in particular there are no fields of size 6, 10, 14, 15 etc.

In this representation it is very easy to add elements:

$$\left(\sum_{i=1}^n c_i \alpha_i \right) + \left(\sum_{i=1}^n d_i \alpha_i \right) = \sum_{i=1}^n (c_i + d_i) \alpha_i$$

but for multiplying them we need to know $\alpha_i \cdot \alpha_j$ for $1 \leq i, j \leq n$.

R From now on we write $+$ for the first operation \circ and \cdot for the second operation \diamond , since we see K as an extension of $\mathbb{Z}/p\mathbb{Z}$.

Let's see whether we can find out more about the multiplicative structure. Remember that for a group G we have $[|G|]a = e$ for any $a \in G$ by the properties of the order of a group. Since K is a field, K^* is a group and it has one element, namely 0, less than K ; thus $|K^*| = p^n - 1$.

Lemma 4.12 Let K be a finite field. The multiplicative group K^* is cyclic.

Thus, for every $a \in K^*$ we have $a^{p^n-1} = 1$.

Let's look at another field beyond $\mathbb{Z}/p\mathbb{Z}$. We know that they must have p^n elements for some p and n - so what about a field with $2^2 = 4$ elements? This should have a basis of size 2, let's use $\alpha_i = 1$ and $\alpha_2 = a$ then $\mathbb{F}_4 = 0, 1, a, a+1$ and we can simply write out the addition table using the vectorspace structure:

+	0	1	a	$a+1$
0	0	1	a	$a+1$
1	1	0	$a+1$	a
a	a	$a+1$	0	1
$a+1$	$a+1$	a	1	0

To write the multiplication table - if possible - we need to know what a^2 is in terms of 1, a and $a+1$. A table of a group has each element exactly once per row and column. So defining $a^2 = a$ conflict with having already entry a in the first entry of this row (see left table below). Using $a^2 = 1$ means that $a \cdot (a+1) = a^2 + a = 1 + a$ - but then the third column (in the middle table) has already $a+1$ in the first entry. Try $a^2 = a+1$ then $a \cdot (a+1) = a^2 + a = (a+1) + a = 1$ and $(a+1) \cdot (a+1) = a^2 + a + a + 1 = a^2 + 1 = (a+1) + 1 = a$.

\cdot	1	a	$a+1$
1	1	a	$a+1$
a	a	a	
$a+1$	$a+1$		

\cdot	1	a	$a+1$
1	1	a	$a+1$
a	a	1	$a+1$
$a+1$	$a+1$		

\cdot	1	a	$a+1$
1	1	a	$a+1$
a	a	$a+1$	1
$a+1$	$a+1$	1	a

The tables show all group properties except for associativity. We could probe this by checking all combinations but that is very cumbersome.

We could do the same that we just did with \mathbb{F}_4 now with \mathbb{F}_8 , but this would take a while (we would now use 1, a and b as a basis). The multiplication table is now 8×8 . So the question arises:

Problem 4.13 How can we get this “automatically”? How do we compute $\alpha_i \cdot \alpha_j$ without a lookup table?

The idea is to use a polynomial ring to represent the field elements. A polynomial ring also spans a vector space - but contrast to the vector space, the multiplication of polynomials is well defined.

The polynomial ring over field K :

$$K[x] = \left\{ \sum_{i=1}^n a_i x^i \mid n \in \mathbb{N}, a_i \in K \right\}, \quad f \in K[x], \quad f = \sum f_i x_i.$$

Let n be the largest integer with $f_n \neq 0$ then $\deg(f) = n$, *leading coefficient* $\text{LC}(f) = f_n$, *leading term* $\text{LT} = f_n x^n$.

Definition 4.14 — Irreducible. A polynomial $f \in K[x]$ is called *irreducible* if $\deg(f) \geq 1$ and it cannot be written as a product of polynomials of lower degree over the same field, i.e. if $u \mid f$ then $u \in K$ or $u = f$ for all $u \in K[x]$ and $\deg(u) \leq \deg(f)$. Otherwise f is *reducible*. Note that this depends on the field K .

■ **Example 4.15**

- $x^2 - 1 = (x+1)(x-1)$ is reducible in $\mathbb{R}[x]$.
- $x^4 + 2x + 1 = (x^2 + 1)^2$ in $\mathbb{R}[x]$ has no roots but is reducible.
- $x^2 + 1$ is irreducible in $\mathbb{R}[x]$ but reducible in $\mathbb{C}[x]$ by $(x-i)(x+i)$.
- $x^3 + 6x^2 + 4$ is irreducible in $\mathbb{Z}/7\mathbb{Z}$

■

We will now look again at $GF(8) = GF(2^3)$. The question arises, what is $a \cdot b$ or $a \cdot a^2 = a^3$ since $b = a \cdot a = a^2$? We chose $a^3 = a + 1$ and then all operations followed by using this equality. This polynomial - $a^3 + a + 1$ - does not factor over $GF(2)$. Other choices we considered, e.g. $a^3 + 1$ do in fact factor and it was exactly by considering these factors, e.g. $(a+1)$ and $(a^2 + a + 1)$ that we derived contradictions, e.g. $(a+1) \cdot (a^2 + a + 1) = a^3 + 1 = 0$ (using $a^3 = 1$). In the end we worked in $GF(2)[a]/(a^3 + a + 1)$ - the polynomial ring over $GF(2)$ modulo the irreducible polynomial $a^3 + a + 1$.

■ **Example 4.16**

We want to compute $a \cdot (a^2 + a) = a^3 + a^2$ and $(a+1) \cdot (a^2 + a) = a^3 + a^2 + a^2 + a = a^3 + a$. For this we use the irreducible polynomial $a^3 + a + 1$ and do a polynomial long division:

$$\begin{array}{r} a^3 + a + 1 \overline{) a^3 + a^2} = 1 \\ \underline{a^3 + a + 1} \\ a^2 + a + 1 \end{array} \qquad \begin{array}{r} a^3 + a + 1 \overline{) a^3 + a} = 1 \\ \underline{a^3 + a + 1} \\ 1 \end{array}$$

■

In general, this construction gives a finite field: Let f be an irreducible polynomial of degree n over $GF(p)$. We define addition and multiplication on

$$GF(p)[x]/f(x)GF(p)[x] = \left\{ \sum_{i=0}^{n-1} a_i x^i \mid a_i \in GF(p) \right\}$$

as addition and multiplication in $GF(p)[x]$ followed by reduction modulo $f(x)$. $\rightsquigarrow GF(p^n)$

Let $g \in GF(p^n)$, f irreducible in $GF(p)$. $\gcd(f, g) = 1$ (The gcd of f and g has to be 1 since f is irreducible) and XGCD computes polynomials h and l with

$$1 = g \cdot h + f \cdot l$$

$$1 \equiv g \cdot h \pmod{f}$$

$$h \equiv g^{-1} \pmod{f}$$

■ **Example 4.17**

The polynomial $f = x^3 + x^2 + 1$ is irreducible over $GF(2)$. What is the inverse of $g = x^2 + 1$ over $GF(2) \pmod{f}$?

$$\begin{array}{r} x^2 + 1 \overline{x^3 + x^2 + 1} = x + 1 \\ \underline{-(x^3 + x)} \\ x^2 + x + 1 \\ \underline{-(x^2 + 1)} \\ x \end{array}$$

$$\Rightarrow x^3 + x^2 + 1 = (x^2 + 1)(x + 1) + x \quad (*)$$

$$\begin{array}{r} x \overline{x^2 + 1} = x \\ \underline{-x^2} \\ 1 \end{array}$$

$$\Rightarrow x^2 + 1 = x \cdot x + 1 \quad (**)$$

$$1 = f \cdot ? + g \cdot ?$$

$$\begin{aligned} 1 &= (x^2 + 1) + x \cdot x && \text{from } (**) \\ &= (x^2 + 1) + [(x^3 + x^2 + 1) + (x^2 + 1)(x + 1)]x && \text{from inserting } (*) \\ &= (x^2 + 1) + (x^3 + x^2 + 1)x + (x^2 + 1)(x + 1)x \\ &= (x^3 + x^2 + 1)x + (x^2 + 1) + (x^2 + 1)(x + 1)x \\ &= (x^3 + x^2 + 1)x + (x^2 + 1)[1 + (x + 1)x] \\ &= (x^3 + x^2 + 1)x + (x^2 + 1)[x^2 + x + 1] \end{aligned}$$

■

Alternative approach:

We know that $a^{p^n} = a$ and $a^{p^2-1} = 1$ for $a \in GF(p^n)$ (LAGRANGE'S Theorem). Thus $a \cdot a^{p^n-2} = a^{p^n-1} = 1$.

So we can compute the inverse of $(x^2 + 1)$ as $x^2 + 1$ in $GF(8)$:

$$(x^2 + 1)^{8-2} = (x^2 + 1)^6 = (x^2 + 1)^4 \cdot (x^2 + 1)^2 = ((x^2 + 1)^2)^2 \cdot (x^2 + 1)^2$$

How do we find irreducible polynomials? We can pick a random polynomial and check if it is irreducible using “RABIN’S test of irreducibility”.

Notation 4.18. We denote \mathbb{F}_q^* as the multiplicative group of \mathbb{F}_q . $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ is a cyclic group and if $\mathbb{F}_q^* = \langle g \rangle$ then the generator g is called a primitive element.

■ **Example 4.19**

$$\mathbb{F}_7^* : \langle 2 \rangle = \{2^0 = 1, 2^1 = 2, 2^2 = 4\}$$


$2^3 = 1$, etc. so $\langle 2 \rangle$ contains only 3 elements, so 2 is not primitive.

$$\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\} = \mathbb{F}_7^*$$

so 3 is a primitive element. ■

Lemma 4.20 Let G be a cyclic group of order n , then the order of any $a \in G$ satisfies:

$$\text{ord}(a) | n$$

 $\text{ord}(a) | n$ states how many times we need to multiply a to receive 1, i. e. $a^n = 1$.

■ **Example 4.21**

- $\text{ord}(4) = 6$ in \mathbb{F}_7^*
 - $\text{ord}(2) = 3$ in \mathbb{F}_7^*
 - $\text{ord}(1) = 1$ in \mathbb{F}_7^*
 - $\text{ord}(6) = 2$ in \mathbb{F}_7^*
-

Lemma 4.22 Let G be a cyclic group of order n , let $\langle g \rangle = G$, and let $l | n$.

$$\text{Then } \text{ord}(g^{n/l}) = l$$

Proof.

$$1, g^{n/l}, g^{2n/l}, g^{3n/l}, \dots, g^{ln/l} = 1$$

so $\text{ord}(g^{n/l})$ is no larger than l and $i \cdot n/l$ for $0 \leq i < n$ is less than n . Because n is minimum for g , l is minimal for n/l . ■

■ **Example 4.23**

$$\langle 3 \rangle = \mathbb{F}_7^*; \text{ord}(3^{6/2} = 3^3) = \text{ord}(6) = 2$$

■

There are multiple elements of order l : all the $g^{i \cdot n/l}$ have order dividing l and if $\gcd(i, l) = 1$ then $\text{ord}(g^{i \cdot n/l}) = l$.

■ **Example 4.24**

$$\mathbb{F}_{19}^* = \langle 2 \rangle, \quad 19 - 1 = 18 = 2 \cdot 3^2$$

$$\{1, 2, 4, 8, -3, -6, 7, -5, 9, -1, -2, -4, -8, 3, 6, -7, 5, 9\} = \langle 2 \rangle$$

$$2 \text{ has } \text{ord}(6), \text{ but } \text{ord}(2^{2 \cdot 18/6}) = 3, \text{ ord}(2^{3 \cdot 18/6}) = 2$$

This means there are $\phi(l)$ elements of order l . ■

5. Diffie-Hellman Key Exchange

5.1 Introduction

The DIFFIE-HELLMAN KEY EXCHANGE is used to secretly exchange keys for an encryption. The key exchange can be explained easily with an example:

A(LICE) and (B)OB want to exchange a key, without letting the eavesdropper E(VE) know. Everybody (including) EVE knows a cyclic group G , a generator g and its order n :

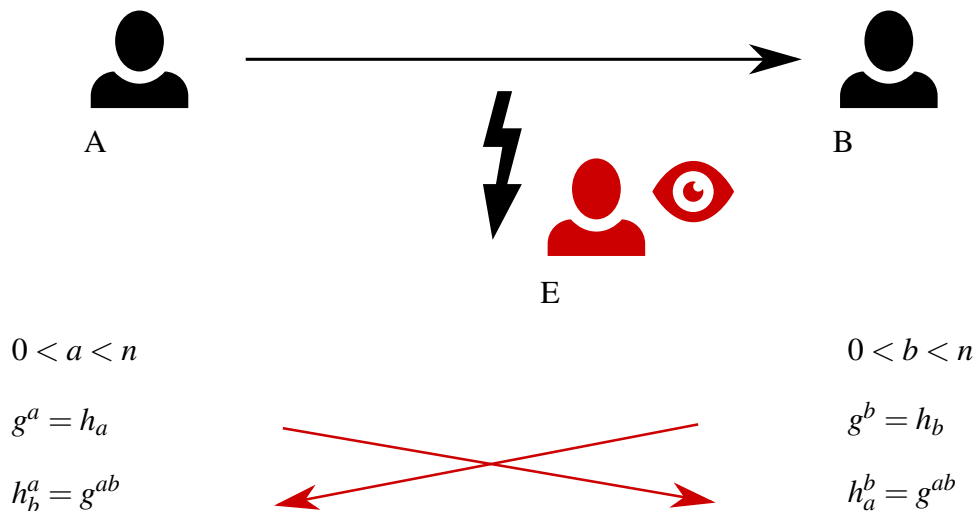


Figure 5.1: Illustration of the DIFFIE-HELLMAN KEY EXCHANGE

A and B both think of a number between 0 and n (this is a and b respectively). Now they compute g^a and g^b and exchange this information (so E(VE) knows g^a and g^b). Then A computes g^{ba} and B computes g^{ab} . So they both now know g^{ab} .

In summary, this means that A computes $h_b^a = (g^b)^a = g^{ba} = g^{ab}$ and B computes

$h_a^b = (g^a)^b = g^{ab}$. So A and B both share g^{ab} , while Eve only knows g , $h_a = g^a$, $h_b = g^b$.

5.1.1 Computational DH Problem (CDHP)

Given g , g^a , g^b compute g^{ab} .

In groups that are good for crypto there are no efficient attacks on the CDHP.

■ **Example 5.1** Examples for these “good groups” are:

- elliptic curves over finite fields
- multipl. groups of finite fields

■

5.1.2 Decisional DH Problem (DDHP)

Given g , g^a , g^b and g^c .

Decide whether $g^c = g^{ab}$.

Proofs for protocols often use DDHP rather than CDHP.

Bad groups would be: $\langle 2 \rangle \subseteq \mathbb{Q}^*$:

$$1, 2, 4, 18, 16 \cdots, 2^i, \dots \quad \text{no reduction}$$

$h_a = 2^a$, E takes log and gets a .

5.1.3 Discrete Logarithm Problem (DLP)

Given g and g^a , compute a .

Solving DLP implies solving CDHP implies solving DDHP. Usually DLP is the best attack we know on DHP, but there is no equivalence.

In browser DH or DHE indicates that DH in finite fields is used.

DHE: ephemeral DH, i. e. use a new key for every connection or for each time interval. Perfectly forward secrecy. Somebody taking over your system at time t should not be able to decrypt any message prior to time t (or $t - t_0$, where t_0 is the time for the ephemeral key).

DH: DIFFIE-HELLMAN with longterm keys.

Crypto parameters choose p to have ≥ 2048 bits and prime, work in \mathbb{F}_p^* , are okay with long-term use; ephemeral is to deal with stuff outside crypto.

A(LICE) and B(OB) use share g^{ab} after running it through a hash function to get a fixed-length string (128 or 256 bits) with good distribution. Cryptographic hash functions are also:

- *pre-image resistant*, i. e. cannot find g^{ab} from $h(g^{ab})$
- *second-pre-image resistant*, i. e. cannot find another pre-image given the first one
- *collision resistant*, i. e. cannot find two strings m_1 and m_2 with $h(m_1) = h(m_2)$.

R We know that m_1 and m_2 exist, but we don't want to be able to compute them.

In DH we use $h(g^{ab})$ as shared key in symmetric crypto, e. g. AES.

5.2 ElGamal Encryption

General parameters: g, n

ALICE'S public key: $h_a = g^a$

ALICE'S secret key: a

Encryption: Pick random $0 < k \leq n$, compute $r = g^k$, $c = h_a^k \cdot m$ (assume $m \in G$), with m the message. We then send (r, c) .

Decryption: $c/r^a = m'$. This works, i. e. $m = m'$ because

$$\frac{c}{r^a} = \frac{h_a^k \cdot m}{(g^k)^a} = \frac{(g^a)^k \cdot m}{(g^a)^k} = m$$

In practice, ELGAMAL is not used like this because $m \notin G$. Instead $c = AES_{h(h_a^k)}(m)$ and m' is computed by first computing $K = h(r^a)$ and then $AES_K(c) = m'$. This corresponds to asymmetric DH.

A has longterm key, sender has K, r as one time keys, but DH uses h_a^K directly as key instead of transmitting m .

5.2.1 ElGamal Signature Scheme

Parameters as above; signature proves that the signer has access to a . Signature is on $h(m)$ not m - fixed length; no algebraic relations.

Sign: Pick **one-time** $0 < k < n$ nonce (= number used only once), compute $r = g^k$, compute $s = k^{-1}(h(m) - a \cdot r) \bmod n$. The signature is (r, s) .

Verify: Does $g^{h(m)}$ equal $h_a^r \cdot r^s$? A valid signature passes verification because:

$$h_a^r \cdot r^s = g^{ar} \cdot g^{k(k^{-1}(h(m)-ar))} = g^{h(m)} \quad \checkmark$$

Anybody knowing a can sign. This becomes a problem if the a being used, becomes known:

We see (r, s) on m , so we can compute

$$s \cdot k = h(m) - a \cdot r \xrightarrow[\text{are known}]{\text{since } s, k, r, h(m)} a = \frac{h(m) - s \cdot k}{r} \bmod n$$

So we can recover A's longterm secret a with just one leaked nonce k . We can also recover a if k is reused (see the homework), we can detect this from seeing repeated r .

R The problem of re-using k , made the first PLAYSTATION open for attacks

This means that ELGAMAL signatures are somewhat fragile. We can work around this by choosing k pseudorandomly, i. e. A has two secrets: a and k_a , compute k as $k = h(k_a, m)$. This gives the same k for repeated m , but the attack on repeated k needs different m 's. If you want a shorter secret have a master secret key s_a and derive $a = h(s_a, \alpha)$ - with α a string "nonce key".

Note that biases in k also lead to breaks via the hidden number problem (we may take about this later), so would need really good randomness for each k - or this construction.

Properties of hash functions are important here because a second pre-image of $h(m)$, i. e. a message m' with $h(m) = h(m')$ can just use the same (r, s) .

If E has colliding messages m_1 and m_2 , where m_1 is innocent, she can ask A to sign m_1 and use that (r, s) as signature on m_2 .

5.3 Pohlig-Hellman Attack

For the DDH we know g, g^a, g^b, g^c , and the problem is, is $g^c = g^{ab}$?

Sometimes we can solve this without solving CDH:

$$\begin{aligned} \mathbb{F}_{19}^* &= \langle 2 \rangle & h_a &= 7, & h_b &= 11 \\ g^c &= 13 \stackrel{?}{=} DH(7, 11) \end{aligned}$$

We can figure out if a was even or odd. $\text{ord}(2) = 18$, so 18 is smallest power of 2 giving 1.

$$\text{If } a = 2a', \text{ then } h_a^9 = 2^{a^9} = 2^{2a'^9} = 2^{18a'} = 1$$

$$\text{else } a = 2a' + 1, \text{ then } h_a^9 = 2^{9(2a'+1)} = 2^9 = 18$$

$$7^9 = 1 \text{ same for } h_b: 11^9 = 1, \text{ so } a \text{ and } b \text{ are both even, so } a \cdot b \text{ is even.}$$

Try 13^9 to see whether it can be $2^{ab} : 13^9 = 18 \implies$ this is not $g^{ab} \implies$ we solved DDH.

This way we can solve the DDH whenever parity of c and ab does not match.

To make statements about probability of solving DDH we work with sequences of triples and try to distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^c) with random c .


Avoid this DDH weakness by picking g to have odd order, so in \mathbb{F}_{19}^* use the subgroup generated by $2^2 = 4$ of order 9. This doesn't solve all our problems, we can check whether g^{ab} and g^c match as third powers:

$7^{18/3} = 7^6 = 1$ shows that a is a multiple of 3, thus $a \cdot b \equiv 0 \pmod 3$. Try whether 13 matches this, here $13^6 = 11$, so another proof that $(7, 11, 13)$ is not a valid DDH triple.

Can we find out more about a and c ? We know:

$$\left. \begin{aligned} a &\equiv 0 \pmod 2 \\ a &\equiv 0 \pmod 3 \end{aligned} \right\} a \equiv 0 \pmod 6$$

we also know $0 < a < 18$, i. e. $a = 6$ or $a = 12$. We can check the two possibilities and get $2^6 = 7$, so $a = 6$.

 This is sort of a “brute-force” approach.

We know $c \equiv 1 \pmod 2$ and $c \not\equiv 0 \pmod 3$. So $c = c_0 + 3 \cdot c_1$, so $2^c = 2^{c_0+3c_1}$, thus $2^{c \cdot 6} = 2^{6(c_0+3c_1)} = 2^{6c_0}$ this gave $13^6 = 11$, so determine c_0 , compare 11 with 2^6 (i. e. $c_0 = 1$) and 2^{12} (i. e. $c_0 = 2$).

$$\text{Here: } 2^6 = 7 \quad 2^{12} = 11 \quad \implies \text{ so } c_0 = 2, c = 2 + 3 \cdot c_1$$

With CRT we know $c \equiv 1 \pmod 2$ $c \equiv 2 \pmod 3 \implies c \equiv 5 \pmod 6$, so it could be 5, 11 or 17.

We could just try these 3 but we want a systematic way of computing DL in groups of composite orders.

$$\begin{aligned}
2^c &= 2^{2+3c_1} \quad , \text{so} \\
2^{3c_1} &= 13/4 = 8 \quad , \text{get } c_1 \text{ by computing} \\
2^{3c_1} &= 2^{6c_1} = 8^2 = 7 \quad \text{and then comparing to} \\
2^0 &= 1, 2^6 = 7 \text{ and } 2^{12} = 11 \implies c_1 = 1
\end{aligned}$$

\implies complete set:

$$c \equiv 1 \pmod{2} \qquad c \equiv 5 \pmod{9} \qquad \rightarrow c = 5$$

This POHLIG-HELLMAN *attack* solves the DLP in all subgroups of prime order and then combines the results using CRT.

■ Example 5.2

$\mathbb{F}_{37}^* = \langle 2 \rangle$, find $a = \log_2 17$,
the group has subgroups of order 2, 3, 4, 9 (and others of non-prime order)

$$17^{18} = 36 \implies a \equiv 1 \pmod{2}$$

$$17^{12} = 26, \text{ so } a \not\equiv 0 \pmod{3}$$

compare with $2^{12} = 26$ and with $2^{2 \cdot 12} = 10 \rightarrow a \equiv 1 \pmod{3}$.

To get $a \pmod{4}$, compute $(17/2)^9 = 27^9 = 36 \Rightarrow a \equiv 1 + 1 \cdot 2 \equiv 3 \pmod{4}$.


Same for $\pmod{9}$: $(17/2)^4 = 27^4 = 10 \implies a = 1 + 2 \cdot 3 \equiv 7 \pmod{9} \implies a \equiv 7 \pmod{36}$. ■

5.3.1 Pohlig-Hellman & DDH

Computation no harder than in largest prime order group and factors weaken DDH \implies work with g of prime order l ($n = l$). New problem: solve DLP in $\langle g \rangle$ of prime order l .

Approaches:

- **Brute Force Search:** Takes at most l steps, on average done after $l/2$.
- **Randomized Brute Force:** (if E suspects that A chooses a in upper part of interval). Attack h_a^b for some known b , i. e. compute DL of $h_a^b = g^{ab}$, get ab , get a by dividing by b modulo l . E can turn this into multi-target attack by searching for $h_a, h_a^{b_1}, h_a^{b_2}, h_a^{b_3}, \dots$ for known b_i ; each would give a after division. But each costs an exponensiation while a step in brute force costs a multiplication. Make the multiple targets cheaper by going after $1, h_a, h_a^2, h_a^3, h_a^4, \dots, h_a^m$. Compare to $1, g, g^2, g^3, g^4, \dots, g^k$ until we find a match. Birthday paradox says that after roughly \sqrt{l} tries we get a collision, so choose $m \approx k \approx \sqrt{l}$

 The birthday paradox states that in a group of only 23 people the probability for a collision of birthdays (two people were born on the same date) is more than 50%.

- **Baby-Step-Giant-Step Algorithm:** Do this systematically, get deterministic algorithm, using $a = a_0 + ma_1$ for $m = \lfloor \sqrt{l} \rfloor$ with $0 \leq a_0 < m$ and $0 \leq a_1 \leq m \underbrace{+1}_{\text{no need}}$.

1. Compute Baby Steps:

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

2. Put pairs (g^i, i) in sorted table

3. Compute $g^{-m} = G$

4. Compute $h_a, h_a G, h_a G^2, \dots$ the Giant-Steps at every result, compare with table

5. Once a match is found, i. e. $g^i = h_a \cdot G^j = h_a \cdot g^{-mj} \iff g^{i+mj} = h_a$, so output $i + mj$.

Running time: Step 2 takes m mults, Step 4 takes at most m mults \implies Total $\leq 2\sqrt{l}$ and on average $1.5\sqrt{l}$ (can randomize to avoid corner cases)

Downside: algorithm needs \sqrt{l} storage. If one trades space for time, e. g. does only $l^{1/3}$ BS, then one needs $\approx l^{2/3}$ GS, so overall time goes up.

Summary: DLP in any group can be solved in $\mathcal{O}(\sqrt{l})$.

Pollard's Rho Algorithm

Probabilistic algorithm, runs in $\sqrt{\frac{\pi}{2}l}$ (average case) steps. Perform a random walk in $\langle g \rangle$, picking powers of g and h_a at random. If $g^i h_a^j = g^k h_a^m$ then $g^{i-k} = h_a^{m-j}$ so $a \equiv (i-k)/(m-j) \pmod{l}$ (if invertible).

Define random walk $f(W_i) = W_{i+1}$, depending only on current position \implies no memory of how we got there.

Expensive but very random

$$f(W_i) = \underbrace{g}_{F(W_i)} \underbrace{h_a}_{G(W_i)} \quad \text{two exponents}$$

Cheaper and fairly random: Make a small table of random steps $g_i h_a^j$ and select next step from this table \implies just one mult per step.

Schoolbook version for POLLARD'S RHO in $\mathbb{F}_p^* \langle g \rangle$. Take $W_a = h_a^r$, r random, update as

$$W_{i+1} = \begin{cases} W_i \cdot g & , W_i \equiv 0 \pmod{3} \\ W_i \cdot h & , W_i \equiv 1 \pmod{3} \\ W_i^2 & , W_i \equiv 2 \pmod{3} \end{cases}$$

That means that if $W_i = g^j h_a^k$ then

$$W_{i+1} = \begin{cases} g^{i+1} h_a^k & , W_i \equiv 0 \pmod{3} \\ g^j h_a^{k+1} & , W_i \equiv 1 \pmod{3} \\ g^{2j} h_a^{2k} & , W_i \equiv 2 \pmod{3} \end{cases}$$

Avoid using memory by using FLOYD'S CYCLIC FINDING METHOD Detect whether your walk has entered a cycle by comparing $W_i \stackrel{?}{=} W_{2i}; W_{i+1} \stackrel{?}{=} W_{2(i+1)}$ etc. Do this as one

fast walk taking 2 Steps at once and one slow walk. At every step compare \implies twice the work, but only 2 group elements to store. Needs more computation and we won't find the first point that's visited twice - but only a small constant worse. Better than schoolbook: make table with random powers, e. g. 2048 steps precomputed, choose these as powers of g only: h_a only for entrance points.

Parallel Pollard Rho

About factor n speed up for n computers, some storage, some communication.

Mark some group elements as “distinguished points”, e. g. elements with top 30 bits equal to 0.

Whenever a walk reaches such a point, it reports the point and the powers of g and h_a to a server, the server sorts and stores these; eventually finds a collision between walks \implies DLP.

In practice: Communicate and store as little information as possible, so stop walk at distinguished point and start a new walk at some h'_a . Report only (DP, r) . Do not compute the power of g , this happens only on the server and only once a collision is found. If DP-property is 30 bits equal 0, then each walk takes about 2^{30} steps.

Total number of DPs reported to the server is roughly $l^{1/2}/2^{30}$ (or divide by 2^n if n bits are 0 for DPs).

\rightarrow works in every group not just \mathbb{F}_q^* .

The next attack is:

5.4 Index Calculus

Does use the structure of \mathbb{F}_q^* , works in some groups but not in all.

In $\mathbb{F}_p^* 10 = 2 \cdot 5$ ($p > 10$) is meaning full, so idea is to lift elements from \mathbb{F}_p to \mathbb{Z} and factor them there; or to lift from \mathbb{F}_{2^n} to $\mathbb{F}_2[x]$ using $\mathbb{F}_{2^n} \cong \mathbb{F}_2[x]/f$ with f irreducible of degree n .

R The “lift” operation can be seen as the opposite of the “reduce” (e. g. taking a large integer mod 17, would result in a number between 0 and 16) operation.

In \mathbb{F}_{p^n} can lift to \mathbb{F}_p or $\mathbb{F}_p[x]$; depending on size of p and n pick one or the other.

Take DLP: g, h find $\log_g h = a$

■ Example 5.3

$h = 10$, so $h = 2 \cdot 5$ and thus $a = \log_g 2 + \log_g 5$ because $h = g^a = 2 \cdot 5 = g^{\log_g 2} \cdot g^{\log_g 5}$ assuming g generates the entire group. Else we can use factorizations involving primes that are powers of g . ■

Why does turning one DLP into two DLPs help?

Idea: find DLs of all small primes by getting lots of relations of the form:

$$g^{b_i} = \prod p_j^{e_{ij}}$$

where the p_i are in $\mathcal{P} = \{\text{primes smaller than } B\}$, B chosen appropriately. \mathcal{P} is called the

factorbase

$$\begin{aligned} b_i &= e_{i1} \log_g p_{11} + e_{i2} \log_g p_{12} + \cdots + e_{im} \log_g p_{1m} \\ &\vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \vdots \\ b_k &= e_{k1} \log_g p_{k1} + e_{k2} \log_g p_{k2} + \cdots + e_{km} \log_g p_{km} \end{aligned}$$

This is a linear system of equations in the unknowns $\log_g p_{ij}$, use all primes in \mathcal{F} for all expressions, i. e. $g^{b_i} = \prod_{j=1}^m p_j^{e_{ij}}$ using $e_{ij} = 0$ if a prime does not appear. We have m unknowns $\log_g p_j$, so we need $\geq m$ equations.

■ **Example 5.4**

$p = 107, \mathbb{F}_{107}^* = \langle 2 \rangle, h = 57$, find $\log_2 57$. Choosing $\mathcal{F} = \{2, 3, 5, 7\}$ know already $\log_2 2 = 1$

$$\begin{aligned} 2^7 &= 21 = 3 \cdot 7 \\ 2^{11} &= 2 \cdot 3 \cdot 5 \\ 2^{77} &= 3^2 \cdot 7 \end{aligned}$$

$$\begin{array}{c} \log 3 \quad \log 5 \quad \log 7 \\ \left(\begin{array}{ccc|c} 1 & 0 & 1 & 7 \\ 1 & 1 & 0 & 11 \\ 2 & 0 & 1 & 77 \end{array} \right) \implies \begin{array}{c} \log 3 \quad \log 5 \quad \log 7 \\ \left(\begin{array}{ccc|c} 1 & 0 & 1 & 7 \\ 0 & 1 & -1 & 4 \\ 0 & 0 & -1 & 63 \end{array} \right) \implies \begin{array}{c} \log 3 \quad \log 5 \quad \log 7 \\ \left(\begin{array}{ccc|c} 1 & 0 & 0 & 70 \\ 0 & 1 & 0 & 47 \\ 0 & 0 & 1 & 43 \end{array} \right) \end{array}$$

In the last step we used: entries in this matrix are exponents of g , so we compute mod $\text{ord}(g) = 106$.

Now we know

$$3 = 2^{70}; \quad 5 = 2^{47}; \quad 7 = 2^{43}$$

Note: We haven't used h here, so this computation can be used to quickly attack any DLP in \mathbb{F}_{107}^* , at the cost of finding one relation involving h with factors over \mathcal{F} . Here:

$$\begin{aligned} h &= 57 = 3 \cdot 19 \not\in \mathcal{F} \\ h \cdot g &= 57 \cdot 2 = 7 \in \mathcal{F} \\ \implies \log_g h &= \log_g 7 - 1 = 43 - 1 = 42 \end{aligned}$$

■

A number is *smooth* if it factors into small $\approx \in \mathbb{F}$ primes. Work on index calculation to choose g^{b_i} more likely to be smooth.

Definition 5.5 — L-Notation. In size N set:

$$L_N(\alpha, c) = \exp(c(\log N)^\alpha (\log \log N)^{1-\alpha})$$

■ **Example 5.6**

- $\alpha = 1, L_N(1, c) = \exp(c(\log N)^\alpha) = N^c$

- $\alpha = 0, L_N(0, c) = \exp(c \log \log N) = \log N^c$

α between 0 and 1 interpolates between polynomial and exponential time. ■

Note: These are relative to size of N , i. e. relative to $\log N$.

Algorithm shown so far runs in time $L_q(1/2, c)$, c depending on q . Current research $L_q(1/3, c)$ in general and $L_q(1/4, c)$ and even less for small p_i , i. e. $p \in \{2, 3\}$ and $q = p^n$.

5.4.1 Logjam

Can downgrade TLS to using DH in \mathbb{F}_p^* with $\log_2 p \approx 512$.

There are very few different primes in use, so one can just precompute the relations for each of those primes; the attack per target is so fast that one can mount a man-in-the-middle attack.

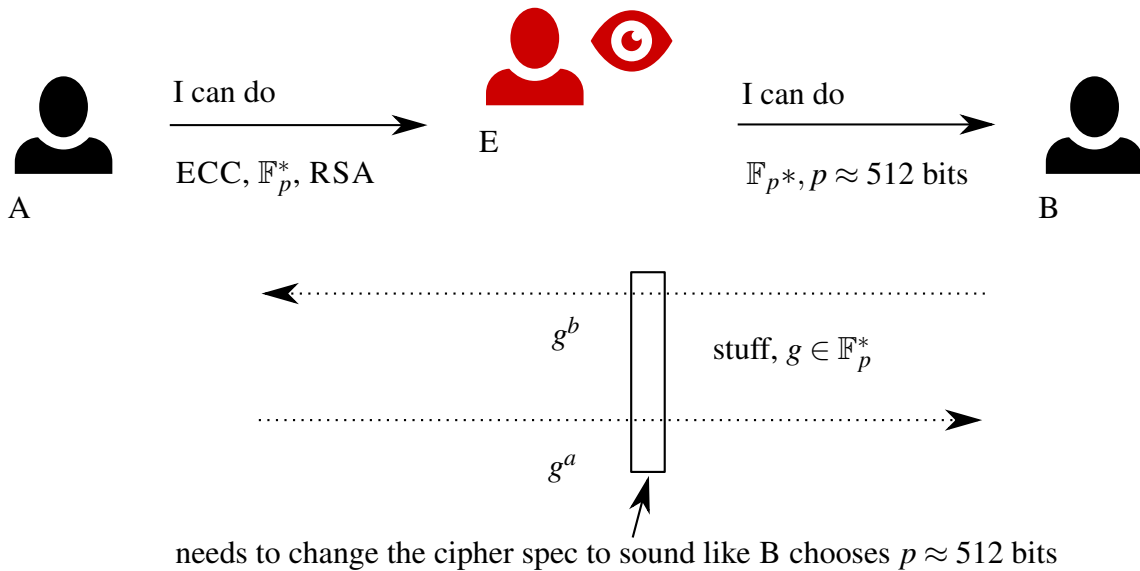


Figure 5.2: Man-in-the-Middle Attack

The change of the cipher spec is secured by a cryptographic checksum. In order to fake this, E needs to have the DL of g^b . WEAKDH.ORG has done this for 512; 768 is within reach. Maybe big agencies can do 1024 bits. Interesting about larger sizes that are still used without downgrade.

Note: $L_q(1/3, c)$ does not depend on group size l , so balance sizes in $l|q - 1$ so that $\sqrt{l} \approx L_q(1/3, c)$.

KEYLENGTH.ORG:

$\log_2 l$	$\log_2 q$	security level
160	1024	80 bits, weak
192	2048	96 bits, weak
256	3072	128 bits, OK

Interesting for signatures where computation is done $\text{mod } l$ as well. DSA digital signature algorithm (NIST/NSA 1992) uses g as generator of small subgroup; arranges signature equation so that $R = g^k$ is lifted to \mathbb{Z} and $r \equiv R \text{ mod } l$, so (r, s) has length $2 \log_2 l$.

Doesn't help much in DH, so many suggestions use \mathbb{F}_p , where $p = 2p' + 1$, p' is prime, so use $l = p'$ and s. t. 2 has order p' .

Common groups, e. g. OAKLEY PRIMES are fixed in protocols - no need for this from mathematical stand point but safer for implementation.

Index

A

Abelian Group 17

B

Baby-Step-Giant-Step Algorithm 30

C

Caesar Cipher 5

Characteristic 18

Chinese Remainder Theorem 14

Computational DH Problem (CDHP) .. 26

Cryptanalysis 5

Cryptography 5

Cryptology 5

D

Decisional DH Problem (DDHP)..... 26

Diffie-Hellman Key Exchange 25

Discrete Logarithm Problem..... 26

Distinguished Points 31

E

ElGamal Encryption 27

ElGamal Signature Scheme..... 27

Euclidean Algorithm 10

Extension Degree 18

F

Factorbase 32

Fast Fourier Transformation 15

Finite Field 17

Floyd's Cyclic Finding Method 30

G

Galois Field 19

I

Index Calculus 31

Irreducible 21

Isomorphism..... 19

K

Keyed Encryption 6

L

L-Notation 32

Lagrange's Theorem 11

Logjam 33

O

Oakley Primes 34

P

Parallel Pollard Rho 31

Pohlig-Hellman Attack 28

Pollard's Rho Algorithm 30

Polynomial Ring 21

Prime Field 19

Private Key 6

Public Key 6

R

Rabin's Test of Irreducibility 23

Reducible 21

RSA 13

S

Square and Multiply 10

Subfield 18