SECURITY & CODES
PART II - SECURITY

JAAP TOP

Contents

1. Advanced Encryption Standard

In 2001 the American *National Institute of Standards and Technology* (NIST) announced a new data-encryption system for safeguarding sensitive information: the AES (Advanced Encryption Standard). This replaces the older DES (Data Encryption Standard), which has been used since 1976. By now, the AES is used internationally for a lot of data transmission. For example, the American National Security Agency (NSA) recommends AES, and therefore it is used by (among others) the American government for all its secret and top-secret data transmission. Various data compression algorithms (e.g., WinRAR and WinZip) offer possibilities to secure data by means of the AES.

The AES is a special case of the system 'Rijndael', invented in Leuven by Vincent Rijmen and Joan Daemen. A short mathematical explanation of it was offered in 2002 by professor H.W. Lenstra from Leiden University. His text can be found at the end of this section; we now supplement it with additional details.

Date: Groningen, August 2009 to October 2012 (English translation 2015).

In the AES, the data one wishes to protect is first subdivided into blocks ('states') consisting of 16 bytes (so, a state contains $16 \cdot 8 = 128$ bits). Furthermore, there is a secret 'key', which is also a block consisting of 16 bytes. For some applications such as 'top-secret' data, keys of 24 or even 32 bytes are used. By means of this key a bijection from the set of 'states' to the set of 'states' is constructed. Applying this bijection to the 'states' we want to protect, results in new data. Reconstructing the original data from it is easy using the secret key, however without the key it appears extremely difficult to invert the bijection used, and thus to decrypt the protected data.

Bytes. The space of all possible bytes is by definition $\mathbb{F}_2^8$. Just as for cyclic codes of length 8, we sometimes identify this space with $R_8 := \mathbb{F}_2[x]/(x^8+1)$. Compared with the case of cyclic codes, this is done in reverse here: identify $(a_7, a_6, a_5, \ldots, a_1, a_0)$ with $a_7x^7 + a_6x^6 + \ldots + a_1x + a_0$. Multiplication by x in R_8 is therefore the same as 'shifting everything one position to the left' in $\mathbb{F}_2^8$. The map

$$\lambda \;:\; R_8 \to R_8,\;\; f \mapsto (x^4+x^3+x^2+x+1)f + x^6+x^5+x+1 \bmod (x^8+1)$$

is a bijection on the space of bytes: namely, $\lambda = \tau_g \circ \mu_h$, with τ_g the translation over $g = x^6+x^5+x+1$ and μ_h the multiplication (modulo x^8+1) by $h = x^4+x^3+x^2+x+1$. Hence λ is a bijection provided both τ_g and μ_h are. Every translation is indeed bijective (with translation over the opposite vector as inverse; so in this case $\tau_g^{-1} = \tau_g$). It remains to show that μ_h is invertible.

For this we note that in $\mathbb{F}_2[x]$ one has $(1+x)^2 = 1+x^2$, and hence $(1+x)^4 = (1+x^2)^2 = 1+x^4$, so finally $(1+x)^8 = (1+x^4)^2 = 1+x^8$. So $1+x^8$ factors as the eighth power of $1+x$. This $1+x$ is no factor of $h = x^4+x^3+x^2+x+1$, since otherwise $h(1) = 0$. So h and $s := 1+x^8$ have no common factor (except 1).

The extended Euclidean algorithm will therefore provide a linear combination of h and s equal to 1. Explicitly:

$$\begin{array}{rcrcc}
0\cdot h & + & 1\cdot s & = & x^8+1 \\
1\cdot h & + & 0\cdot s & = & x^4+x^3+x^2+x+1 \\
x^4\cdot h & + & 1\cdot s & = & x^7+x^6+x^5+x^4+1 \\
(x^3+x^4)\cdot h & + & 1\cdot s & = & x^3+1 \\
(1+x^4+x^5)\cdot h & + & x\cdot s & = & x^3+x^2+1 \\
(1+x^3+x^5)\cdot h & + & (1+x)\cdot s & = & x^2 \\
(1+x+x^5+x^6)\cdot h & + & x^2\cdot s & = & x^2+1 \\
(x+x^3+x^6)\cdot h & + & (1+x+x^2)\cdot s & = & 1.
\end{array}$$

Check for yourself how these lines were obtained; the first and second are evident, and subsequent lines are linear combinations of the two directly preceding ones, in such a way that the right-hand-side has degree below the expression two lines above it. The bottom line, considered modulo $s = x^8+1$, yields that $(x+x^3+x^6)h \equiv 1 \bmod (x^8+1)$.

Conclusion: $\mu_{x+x^3+x^6}$ satisfies

$$\mu_{x+x^3+x^6} \circ \mu_h(f) = (x + x^3 + x^6)hf \bmod (x^8 + 1) = f \bmod (x^8 + 1)$$

and also $\mu_h \circ \mu_{x+x^3+x^6}(f) = f \bmod (x^8 + 1)$, for every f.

Hence μ_h is invertible, with $\mu_h^{-1} = \mu_{x+x^3+x^6}$. This provides the inverse for λ as well, namely

$$\lambda^{-1} = \mu_h^{-1} \circ \tau_g = \mu_{x+x^3+x^6} \circ \tau_g.$$

So this map sends an f to $(x + x^3 + x^6)(f + g) \bmod (x^8 + 1)$. Since $(x+x^3+x^6)(x^6+x^5+x+1) \equiv x^2+1 \bmod (x^8+1)$, we have a simpler description:

$$\lambda^{-1}(f) = (x + x^3 + x^6)f + x^2 + 1 \bmod (x^8 + 1).$$

Calculating in a structure such as $\mathbb{F}_2[x]/(x^8 + 1)$ may be done, using the Magma package, as follows.

```
F2:=GF(2);
P<X>:=PolynomialRing(F2);
I:=ideal<P | X^8+1 >;
R<x>:=P/I;
la:=function(f)
      return R!((1+x+x^2+x^3+x^4)*R!f+x^6+x^5+x+1);
    end function;
g:=1+x^5; g;
la(g);
la(la(g));
la(la(la(g)));
la(la(la(la(g))));
```

A second bijection on bytes we now discuss, uses modular arithmetic with polynomials as well. In this case not modulo $x^8 + 1$ in analogy with coding theory, but modulo m, where

$$m = x^8 + x^4 + x^3 + x + 1.$$

Calculating in $\mathbb{F}_2[x]/(m)$ works as expected: $f \equiv g \bmod (m)$ means that $f+g$ is a multiple of m. In this way, using division with remainder by m, every polynomial is equivalent to a polynomial of degree < 8. And if f, g both have degree < 8, then $f \equiv g \bmod (m)$ if and only if $f = g$. So $\mathbb{F}_2[x]/(m)$ can be identified with the space of all bytes, and multiplication modulo (m) yields an operation on this space.

There is a big difference between multiplication modulo (x^8+1) and modulo (m). The reason for this is the fact that m is irreducible: m is not a multiple of any polynomial of positive degree < 8. This can be verified using a rather long and boring computation, or alternatively, using a system such as Magma, and typing

```
F2 := GF(2);
P<x> := PolynomialRing(F2);
Factorization(x^8+x^4+x^3+x+1);
```

This property has a strong consequence: take any $g \in \mathbb{F}_2[x]$ of degree < 8, and $g \neq 0$. Then g and m have no common factor $\neq 1$. The extended Euclidean algorithm therefore yields a linear combination $pg + qm = 1$, for certain polynomials p, q. Modulo m this means that p is an inverse of g. So every $g \neq 0$ in $\mathbb{F}_2[x]/(m)$ has an inverse.

So in $\mathbb{F}_2[x]/(m)$ one can not only add/substract and multiply, but also divide (=multiply by the inverse). Note that such an inverse is unique: is $pg \equiv p'g \equiv 1 \bmod (m)$, then $g(p + p') \equiv 0 \bmod (m)$, and multiplying this by p shows $p + p' \equiv 0 \bmod (m)$ which means $p \equiv p' \bmod (m)$. The inverse of g is denoted g^{-1}.

So it holds that $\mathbb{F}_2[x]/(m)$ is a field, consisting of $2^8 = 256$ elements. We also write $\mathbb{F}_{256}$ for this field.

The map

$$\sigma \;:\; \mathbb{F}_2^8 \to \mathbb{F}_2^8,\quad f \mapsto \begin{cases} \lambda(0) = x^6 + x^5 + x + 1 & \text{if } f = 0; \\ \lambda((f \bmod (m))^{-1}) & \text{otherwise} \end{cases}$$

in the AES is called the *S-box*. It is a bijection on the space of bytes, because the map $\mathbb{F}_{256} \to \mathbb{F}_{256}$ sending 0 to 0 and inverting all other elements, is its own inverse, and the composition of it with the also invertible λ yields σ.

From Lenstra's text on this subject one may conclude, that the order of σ (the minimal number $n > 0$ such that applying σ n times yields the identity) equals $2 \cdot 3^4 \cdot 29 \cdot 59 = 277182$. Nevertheless, σ is just a composition of a bijection of order 2 (namely, inversion in $\mathbb{F}_{256}$) and a bijection of order 4 (namely λ). Determining the order of σ is without a computer quite cumbersome; Using, e.g., Magma or Maple or Mathematica turns it into a pleasant exercise!

A natural question is why the polynomial $h = x^4 + x^3 + x^2 + x + 1$ (with the necessary property $h(1) \neq 0$) and the polynomial $g = x^6 + x^5 + x + 1$ and the (necessarily irreducible) polynomial $m = x^8 + x^4 + x^3 + x + 1$ are used in defining the S-box, instead of other polynomials. A partial answer to this question is given in the bachelor's thesis (written in Dutch) of Petra Klooster (2014), see `http://fse.studenttheses.ub.rug.nl/11887/`.

Words. A word is by definition a tuple $w = (b_0, b_1, b_2, b_3)$ in which the b_j are bytes. The map σ described above, provides a bijection on words which we denote by σ as well:

$$\sigma(w) = \sigma(b_0, b_1, b_2, b_3) := (\sigma(b_0), \sigma(b_1), \sigma(b_2), \sigma(b_3))\,.$$

A second operation on words (which in fact will only be used on parts of the secret key) is called ξ; it is given as

$$\xi(w) = \xi(b_0, b_1, b_2, b_3) := (\sigma(b_1), \sigma(b_2), \sigma(b_3), \sigma(b_0)).$$

So the map ξ can be regarded as a shift on the 4 bytes in a word, followed by the map σ.

As an alternative description: write $w = (b_0, b_1, b_2, b_3)$ as a polynomial $b_0 + b_1 y + b_2 y^2 + b_3 y^3$, then $\xi(w) = \sigma(y^3 w \bmod (y^4 + 1))$.

In a similar way we define bijections μ, ν on words: take $(x, 1, 1, x + 1) \in (\mathbb{F}_2[x]/(m))^4$. Regard this as the element

$$c := x + y + y^2 + xy^3 + y^3 \in (\mathbb{F}_2[x]/(m))\,[y]/(y^4 + 1).$$

So, here we view a word as a polynomial

$$c_0(x) + c_1(x)y + c_2(x)y^2 + c_3(x)y^3$$

in the variables x and y, with coefficients in $\mathbb{F}_2$. Such polynomials we multiply, with the agreement that powers of y are considered modulo $y^4 + 1$. To put it differently: we agree that $y^n = y^m$ as soon as $m \equiv n \bmod 4$. And considering the polynomials $c_j(x) \in \mathbb{F}_2[x]$ we have here, we agree to compute with them modulo $m = x^8 + x^4 + x^3 + x + 1$. Under these conditions, the map μ on words, defined for any word $w = c_0(x) + c_1(x)y + c_2(x)y^2 + c_3(x)y^3$, is defined by

$$\mu(w) := c \cdot w.$$

With

$$d := x^3 + x^2 + x + x^3 y + y + x^3 y^2 + x^2 y^2 + y^2 + x^3 y^3 + xy^3 + y^3$$

one defines similarly the map ν on words by

$$\nu(w) := d \cdot w.$$

An important observation is here, that μ and ν are indeed bijections on the set of words. This is a simple consequence of the fact that

$$c \cdot d \equiv 1 \bmod (y^4 + 1),$$

which is easily verified.

In $\mathbb{F}_2[x, y]/(m, y^4 + 1)$ (so, the polynomials in x and y considered modulo m as well as modulo $y^4 + 1$) it holds that $c^4 = d^4 = 1$. This implies $\mu^{-1} = \mu^3 = \nu$ and $\nu^{-1} = \nu^3 = \mu$.

Example: consider the word

$$w = \text{ 00010010 00100100 01001000 10000001}.$$

As a tuple of four polynomials this is $(x^4 + x, x^5 + x^2, x^6 + x^3, x^7 + 1)$ and as a polynomial in x and y it is

$$w = x^4 + x + x^5 y + x^2 y + x^6 y^2 + x^3 y^2 + x^7 y^3 + y^3.$$

By definition $\xi(w) = \sigma(x^5 + x^2, x^6 + x^3, x^7 + 1, x^4 + x)$ and this is computed by the action of σ on bytes. To do this manually, the extended

Euclidean algorithm needs to be done four times, followed by four times the map λ on bytes. Evidently this is rather elaborate. Using software such as Magma and the code we used to compute the map λ, it appears as follows:

```
sigma := function(f)
           if f eq 0 then
             return la(0);
           else
             I:=ideal<P | X^8+X^4+X^3+X+1 >;
             R<x>:=P/I;
             return la(P!((R!(f))^(-1)));
           end if;
         end function;
sigma(X^5+X^2);
```

In this way we find

$$\xi(w) = (x^5 + x^4 + x^2 + x, x^6 + x^4 + x, x^3 + x^2, x^7 + x^6 + x^3 + 1),$$

or, expressed as a sequence of zeros and ones,

$$\xi(w) = \text{ 00110110 01010010 00001100 11001001.}$$

The image of w under the map μ can be expressed, using Maple, as a polynomial in x and y as follows (we show at the same time how to verify that the polynomials c, d satisfy $c \cdot d \equiv 1 \bmod (y^4 + 1)$).

```
F2 := GF(2);
P<X,Y> := PolynomialRing(F2,2);
I:=ideal<P | [X^8+X^4+X^3+X+1,Y^4+1] >;
R<x,y>:=P/I;
c:= x+y+y^2+x*y^3+y^3;
d:=x^3+x^2+x+x^3*y+y+x^3*y^2+
   x^2*y^2+y^2+x^3*y^3+x*y^3+y^3;
c*d;
mu := function (w)
        return c*w;
      end function;
mu(x^4+x+x^5*y+x^2*y+x^6*y^2+x^3*y^2+x^7*y^3+y^3);
```

Check for yourself that this results in

$$\mu(w) = 10000001\ 00000011\ 00111110\ 01000011.$$

States. A 'state' is by definition a tuple $s = (w_0, w_1, w_2, w_3)$ consisting of four words w_j. The maps μ, ν and σ which we already know on words, yield maps from states to states by applying them coordinatewise.

As an example,

$$\sigma(w_0, w_1, w_2, w_3) = (\sigma(w_0), \sigma(w_1), \sigma(w_2), \sigma(w_3)).$$

Another quite simple operation on states is the translation, in cryptography also called 'blinding', and in computer science 'xor'. Given a fixed state s, we denote the translation over s by τ_s. On any state x it is given by

$$\tau_s(x) = x + s.$$

Observe that doing τ_s twice results in the original state. In other words, τ_s equals its inverse: $\tau_s^{-1} = \tau_s$.

All operations considered on states so far, are in fact maps on the four words in a state separately. To make the system more complex, also some map 'mixing' the four words in a state is needed. This is done by means of a map called ρ. To define it, we write the state s as

$$s = (w_0, w_1, w_2, w_3)$$

for words w_j. Consider these words as column vectors

$$w_0 = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}, \quad w_1 = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}, \quad w_2 = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}, \quad w_3 = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}.$$

Here the a_j, b_j, c_j and d_j are bytes. In this way the state s is regarded as a 4×4 matrix of bytes, and its columns are the four words in s. Now define

$$\rho(s) = \rho\left(\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{pmatrix}\right) := \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 & a_2 \\ c_3 & d_3 & a_3 & b_3 \\ d_4 & a_4 & b_4 & c_4 \end{pmatrix}.$$

It is easy to verify that $\rho \circ \rho \circ \rho \circ \rho = id$, hence $\rho^{-1} = \rho^3$.

The key. The secret key used for encrypting and decrypting messages with the AES, is a state $k = (w_0, w_1, w_2, w_3)$ consisting of four (secret) words w_j. Since 2^8 different bytes exist, there are 2^{32} different words and therefore $2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456$ possible keys. This makes the probability of finding it by (repeatedly) guessing, negligible.

Once a key has been agreed, the system begins by expanding the tuple $w_0, \ldots, w_3$ into a sequence $w_0, w_1, \ldots\ldots, w_{42}, w_{43}$, in which the w_j for $j \geq 4$ are determined as follows:

$$w_j := \begin{cases} \xi(w_{j-1}) + w_{j-4} + x^{(j-4)/4} \bmod (m, y^4+1) & \text{if } j \equiv 0 \bmod 4; \\ w_{j-1} + w_{j-4} & \text{otherwise.} \end{cases}$$

Finally the AES. The sequence of words (w_j) constructed above yields for $j = 0, \ldots, 10$ the states

$$k_j := (w_{4j}, w_{4j+1}, w_{4j+2}, w_{4j+3}).$$

So $k_0 = k$ is the initial key and the other k_j have been constructed from it. Encryption according to the AES with key k is defined as the bijection

$$\epsilon_k \ : \ \{\text{states}\} \longrightarrow \{\text{states}\}$$

given by

$$\epsilon_k = \tau_{k_{10}}\rho\sigma\tau_{k_9}\mu\rho\sigma\tau_{k_8}\mu\rho\sigma\tau_{k_7}\mu\rho\sigma\tau_{k_6}\mu\rho\sigma\tau_{k_5}\mu\rho\sigma\tau_{k_4}\mu\rho\sigma\tau_{k_3}\mu\rho\sigma\tau_{k_2}\mu\rho\sigma\tau_{k_1}\mu\rho\sigma\tau_{k_0}.$$

This indeed defines a bijection since each of the involved maps ρ, τ_{k_j}, σ and μ is bijective.

It is easy to describe the inverse of ϵ_k, the map that decrypts an encrypted message, in terms of the known inverses $\rho^{-1} = \rho^3$, $\tau_{k_j}^{-1} = \tau_{k_j}$, σ^{-1} (the latter can be described in terms of the maps λ^3 and inversion in $\mathbb{F}_2[x]/m$), and $\mu^{-1} = \nu = \mu^3$.

Leiden professor H.W. Lenstra summarised all of this in one page, as follows:

Rijndael for algebraists H. W. Lenstra, Jr. April 8, 2002

This page deals only with Rijndael with block length 128 and key length 128.

Bytes. A *bit* is an element of $\mathbf{F}_2 = \mathbf{Z}/2\mathbf{Z}$. Eight bits form one *byte*. The space $\mathbf{F}_2^8$ of all bytes is identified with $\{f \in \mathbf{F}_2[X] : \deg f < 8\}$ by $(b_7b_6b_5b_4b_3b_2b_1b_0) = \sum_{h=0}^{7} b_h X^h$. Define the affine map $\lambda\colon \mathbf{F}_2^8 \to \mathbf{F}_2^8$ by $\lambda(f) \equiv (X^4+X^3+X^2+X+1)\cdot f + X^6+X^5+X+1 \bmod (X^8+1)$. The inverse $\lambda^{-1} = \lambda^3$ is given by $\lambda^{-1}(f) \equiv (X^6+X^3+X)\cdot f + X^2+1 \bmod (X^8+1)$. All other operations on $\{f \in \mathbf{F}_2[X] : \deg f < 8\}$ will be done not mod X^8+1 but mod $m = X^8+X^4+X^3+X+1$, so that $\mathbf{F}_2^8$ becomes identified with the field $\mathbf{F}_{256} = \mathbf{F}_2[X]/(m)$. Define the map $\sigma\colon \mathbf{F}_{256} \to \mathbf{F}_{256}$ by $\sigma(a) = \lambda(a^{254})$; here $a^{254} = a^{-1}$ for $a \neq 0$. The cycle lengths of σ are 2, 27, 59, 81, and 87, and $\sigma^{-1} = \sigma^{277181}$ is given by $\sigma^{-1}(a) = (\lambda^{-1}(a))^{254}$.

Words. Four bytes form one *word*. The map from the space $\mathbf{F}_{256}^4$ $(= \mathbf{F}_2^{32})$ of all words to itself sending $(a_i)_{i=0}^3$ to $(\sigma(a_i))_{i=0}^3$ is again denoted by σ. The map $\xi\colon \mathbf{F}_{256}^4 \to \mathbf{F}_{256}^4$ is defined by $\xi((a_i)_{i=0}^3) = (\sigma(a_{i+1}))_{i=0}^3$ (indices mod 4). Write $c = (X, 1, 1, X+1)$ and $d = (X^3+X^2+X, X^3+1, X^3+X^2+1, X^3+X+1)$, and identify $\mathbf{F}_{256}^4$ with $\{g \in \mathbf{F}_{256}[Y] : \deg g < 4\}$ by $(a_0, a_1, a_2, a_3) = \sum_{i=0}^{3} a_i Y^i$. Define μ, $\nu\colon \mathbf{F}_{256}^4 \to \mathbf{F}_{256}^4$ by $\mu(g) \equiv c\cdot g \bmod (Y^4+1)$ and $\nu(g) \equiv d\cdot g \bmod (Y^4+1)$. One has $\nu = \mu^{-1} = \mu^3$.

States. Four words form one *state*. The maps from the space $\mathcal{S} = (\mathbf{F}_{256}^4)^4$ $(= \mathbf{F}_2^{128})$ of all states to itself sending $(w_j)_{j=0}^3$ to $(\mu(w_j))_{j=0}^3$, to $(\nu(w_j))_{j=0}^3$, and to $(\sigma(w_j))_{j=0}^3$ are again denoted by μ, ν, and σ, respectively. Define $\rho\colon \mathcal{S} \to \mathcal{S}$ by $\rho(((a_{i,j})_{i=0}^3)_{j=0}^3) = ((a_{i,i+j})_{i=0}^3)_{j=0}^3$ (indices mod 4). If a state is written as a 4×4-matrix, each column being a word, then ρ shifts the entries in row i cyclically i places to the left $(0 \le i \le 3)$; similarly, $\rho^{-1} = \rho^3$ shifts row i cyclically i places to the right. One has $\rho\sigma = \sigma\rho$. For $s \in \mathcal{S}$, the map $\tau_s\colon \mathcal{S} \to \mathcal{S}$ is defined by $\tau_s(x) = x + s$; one has $\tau_s^{-1} = \tau_s$ and $\mu\tau_s = \tau_{\mu(s)}\mu$.

Key expansion. The *key* space $\mathcal{K}$ equals $\mathcal{S}$. For fixed $k = (w_j)_{j=0}^3 \in \mathcal{K}$, define inductively $w_4, w_5, \ldots, w_{43} \in \mathbf{F}_{256}^4$ by $w_j = w_{j-1} + w_{j-4}$ if $j \not\equiv 0 \bmod 4$ and $w_j = \xi(w_{j-1}) + w_{j-4} + (X^{(j-4)/4}, 0, 0, 0)$ if $j \equiv 0 \bmod 4$, and put $k_l = (w_{4l}, w_{4l+1}, w_{4l+2}, w_{4l+3}) \in \mathcal{S}$ for $0 \le l \le 10$.

Encryption and decryption. Messages are divided in blocks of 128 bits each. Each block belongs to $\mathcal{S}$. Given a key $k \in \mathcal{K}$, a block is encrypted by means of the encryption function $\varepsilon_k\colon \mathcal{S} \to \mathcal{S}$ defined by

$$\varepsilon_k = \tau_{k_{10}}\rho\sigma\tau_{k_9}\mu\rho\sigma\tau_{k_8}\mu\rho\sigma\tau_{k_7}\mu\rho\sigma\tau_{k_6}\mu\rho\sigma\tau_{k_5}\mu\rho\sigma\tau_{k_4}\mu\rho\sigma\tau_{k_3}\mu\rho\sigma\tau_{k_2}\mu\rho\sigma\tau_{k_1}\mu\rho\sigma\tau_{k_0}$$

(nine μ's, ten ρ's, ten σ's, and eleven τ's; composition is from right to left). The corresponding decryption function $\delta_k = \varepsilon_k^{-1}$ is given by

$$\delta_k = \tau_{k_0}\rho^{-1}\sigma^{-1}\tau_{\nu(k_1)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_2)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_3)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_4)}\nu\rho^{-1}\sigma^{-1} \circ$$
$$\circ\, \tau_{\nu(k_5)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_6)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_7)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_8)}\nu\rho^{-1}\sigma^{-1}\tau_{\nu(k_9)}\nu\rho^{-1}\sigma^{-1}\tau_{k_{10}}.$$

2. DH and RSA and ElGamal signatures

In this section we briefly discuss an elementary number theoretic fact. After this we treat three well known applications of it in cryptography: the Diffie-Helman key exchange protocol, the Rivest-Shamir-Adleman public key cryptosystem and the ElGamal digital signatures.

Let $N \neq 0$ be an integer. The group of all *units modulo* N, denoted as $(\mathbb{Z}/N\mathbb{Z})^*$, consists by definition of all classes $a \bmod N = a + N\mathbb{Z}$ which are units modulo N. This means that $b \bmod N$ exists such that

$$(a \bmod N) \cdot (b \bmod N) = 1 \bmod N.$$

Such $b \bmod N$ exists precisely when $\gcd(a, N) = 1$. Namely, the condition of being a unit can be written as the existence of integers b, c satisfying

$$ba + cN = 1,$$

and the extended Euclidean algorithm shows that these exist (and are easily found!) precisely when $\gcd(a, N) = 1$.

The number of elements in the group $(\mathbb{Z}/N\mathbb{Z})^*$ is denoted by $\varphi(N)$. The map φ, which assigns to every integer $\neq 0$ a positive integer, is called the Euler-φ-function or sometimes the Euler-totient-function.

In any finite group G it holds that if $n = \#G$ and $g \in G$, then g^n equals the unit element of G. A proof of this assertion can be found in essentially every introductory text on the theory of groups. In particular, for all $a \bmod N \in (\mathbb{Z}/N\mathbb{Z})^*$ one has

$$a^{\varphi(N)} \equiv 1 \bmod N.$$

A proof for this, using the commutativity of multiplication in $(\mathbb{Z}/N\mathbb{Z})^*$, runs as follows. Write P for the product of all elements in $(\mathbb{Z}/N\mathbb{Z})^*$. Then also P is an element of this group. Now

$$\begin{aligned} P &= \prod_{b \bmod N \in (\mathbb{Z}/N\mathbb{Z})^*} (b \bmod N) \\ &= \prod_{b \bmod N \in (\mathbb{Z}/N\mathbb{Z})^*} (ab \bmod N) \quad = (a^{\varphi(N)} \bmod N) \cdot P, \end{aligned}$$

since multiplication by $a \bmod N$ is a bijection on the group $(\mathbb{Z}/N\mathbb{Z})^*$. Multiplying by the inverse of P then proves the assertion.

We briefly consider two special cases.

If $N = p$ is a prime number, then $\varphi(N) = N - 1$. Incidentally, the converse holds as well: is $\varphi(N) = N - 1$, then N is prime. Namely, since $\varphi(1) = 1$, the condition implies $N \geq 2$. Therefore $0 \bmod N$ is not a unit modulo N, and hence all other $a \bmod N$ have to be units. This means $\gcd(a, N) = 1$ for all a such that $1 \leq a \leq N - 1$, which shows that N is prime.

For a prime number p it turns out that

$$a^{p-1} \equiv 1 \bmod p$$

whenever a is not divisible by p. This is the celebrated "Fermat's little theorem".

For $N = pq$ with p and q two distinct prime numbers, one has $\varphi(pq) = (p-1)(q-1)$. Namely, the a with $1 \le a \le pq$ which do *not* satisfy $\gcd(a, pq) = 1$, are

$$p, 2p, \ldots\ldots, qp \qquad \text{and} \qquad q, 2q, \ldots\ldots, pq.$$

These are $q+p-1$ numbers, hence $\varphi(pq) = pq-p-q+1 = (p-1)(q-1)$.
So for this case

$$a^{(p-1)(q-1)} \equiv 1 \bmod pq$$

for all a satisfying $\gcd(a, pq) = 1$.

Every element $a \bmod N$ in $(\mathbb{Z}/N\mathbb{Z})^*$ has an *order*; this is (a well known definition from group theory) the smallest integer $d > 0$ such that $a^d \equiv 1 \bmod N$. The following properties hold.

Lemma 2.1. *The order of elements* $a \bmod N$, $b \bmod N \in (\mathbb{Z}/N\mathbb{Z})^*$ *satisfies*

(1) $\mathrm{order}(a \bmod N)$ *divides* $\varphi(N)$*;*
(2) *if* $a^m \equiv 1 \bmod N$*, then* m *is a multiple of* $\mathrm{order}(a \bmod N)$*;*
(3) *if* $\mathrm{order}(a \bmod N) = d$ *and* $\mathrm{order}(b \bmod N) = e$ *with* $\gcd(d, e) = 1$*, then* $\mathrm{order}(ab \bmod N) = de$*;*
(4) *for* p *prime and* $d > 0$*, the group* $(\mathbb{Z}/p\mathbb{Z})^*$ *has at most* d *elements of order dividing* d*.*

Proof. (1.) Write $d := \mathrm{order}(a \bmod N)$ and take $e := \gcd(d, \varphi(N))$. Then $e = xd + y\varphi(N)$ for certain integers x, y, hence since $a^d \equiv 1 \bmod N$ and also $a^{\varphi(N)} \equiv 1 \bmod N$, it follows that

$$a^e \equiv 1 \bmod N.$$

Now $e > 0$ and $e|d$ and d is the smallest integer such that $a^d \equiv 1 \bmod N$, hence $d = e$. Since $e|\varphi(N)$, one concludes $d|\varphi(N)$.
(2.) This is the same argument as given in (1.), with the role of $\varphi(N)$ replaced by m.
(3.) Put $c := \mathrm{order}(ab \bmod N)$. Since $(ab)^{de} \bmod N = 1 \bmod N$, (2.) implies $c|de$. But $(ab)^c = a^c b^c$, hence $b^c \bmod N$ is the inverse of $a^c \bmod N$. In particular,

$$\mathrm{order}(a^c \bmod N) = \mathrm{order}(b^c \bmod N).$$

By (2.), this order is a divisor of both d and e, because $a^{cd} \equiv 1 \bmod N$ and $b^{ce} \equiv 1 \bmod N$. Since $\gcd(d, e) = 1$, this order equals 1. In other words,

$$a^c \bmod N = 1 \bmod N = b^c \bmod N.$$

Now (2.) implies $d|c$ and $e|c$, and since $\gcd(d, e) = 1$ this implies $de|c$. We already saw that $c|de$, so $c = de$ follows.

(4.) The elements of order dividing d are zeros (in $(\mathbb{Z}/p\mathbb{Z})^*$) of the polynomial $X^d - 1$. If a_1 up to a_t are pairwise distinct zeros of this polynomial in $(\mathbb{Z}/p\mathbb{Z})^*$, then write

$$X^d - \overline{1} = (X - a_1) \cdot \ldots \cdot (X - a_t)Q$$

for some polynomial Q with coefficients in $\mathbb{Z}/p\mathbb{Z}$ (using mathematical induction with respect to t one can take out the factors $X - a_j$ one by one). Comparing degrees then shows $t \leq d$. □

A legitimate question is, where exactly in the proof of (4.) as given above, one uses that p is prime. This happens when taking out the factors $X - a_j$. Consider as an example $N = 8$. The elements in $(\mathbb{Z}/8\mathbb{Z})^*$ of order dividing 2 are $\overline{1} = 1 \bmod 8$, $\overline{3} = 3 \bmod 8$, $\overline{5} = 5 \bmod 8$ and $\overline{7} = 7 \bmod 8$. We start with the polynomial $X^2 - \overline{1}$. It has all of the above elements as zero. One can factor $X^2 - \overline{1} = (X - \overline{3})(X - \overline{5})$, but also $X^2 - \overline{1} = (X - \overline{7})(X - \overline{1})$. However, after choosing one such factor, it is not true that all other elements of order dividing 2 are zeros of the remaining factor. This *does* hold in the case that $N = p$ is prime.

Lemma 2.1 has an important consequence:

Theorem 2.2. *For p prime, the group $(\mathbb{Z}/p\mathbb{Z})^*$ contains an element $g \bmod p$ with $\mathrm{order}(g \bmod p) = p - 1$.*

Proof. Let $d \geq 1$ be the largest integer occurring as the order of some element of $(\mathbb{Z}/p\mathbb{Z})^*$. Part (1.) of Lemma 2.1 says that $d | \varphi(p) = p - 1$, so in particular $d \leq p - 1$.

Consider any $a \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ and put $e := \mathrm{order}(a \bmod p)$. We claim that $e|d$. Since d occurs as order of an element, take $b \bmod p$ with $\mathrm{order}(b \bmod p) = d$. If e were not a divisor of d, then some prime number ℓ exists, such that e contains more factors ℓ than d, i.e., $e = \ell^m e_1$ and $d = \ell^n d_1$ for integers d_1, e_1, m, n with $m > n \geq 0$ and d_1 not divisible by ℓ. Since $e = \mathrm{order}(a \bmod p)$, it follows that $a^{e_1} \bmod p$ has order ℓ^m. Similarly $b^{\ell^n} \bmod p$ has order d_1. But $\gcd(\ell^m, d_1) = 1$, hence part (3.) of Lemma 2.1 implies that

$$\mathrm{order}(a^{e_1} b^{\ell^n} \bmod p) = \ell^m d_1 > \ell^n d_1 = d,$$

contradicting the definition of d.

We conclude that every element of $(\mathbb{Z}/p\mathbb{Z})^*$ has an order dividing d, and therefore part (4.) of Lemma 2.1 shows $p - 1 \leq d$. Since we already know the reverse inequality, $p - 1 = d$ follows. This is precisely what we wanted to prove. □

Definition 2.3. *For p prime, any $g \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ with $\mathrm{order}(g \bmod p) = p - 1$ is called a* primitive root modulo p.

Theorem 2.2 asserts that primitive roots modulo p exist, for every prime p. The presented proof is a typical example of a nonconstructive existence proof: it does not provide an efficient algorithm for finding such a primitive root. Various proofs of the same theorem, all nonconstructive, may be found on the site `http://www.math.uconn.edu/~kconrad/blurbs/grouptheory/cyclicFp.pdf`, by the American mathematician Keith Conrad.

2.4. **Discrete logarithms.**

Given a prime p and a primitive root $g \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$, we know that for $0 \le i < j < p-1$ the powers $g^i \bmod p$ and $g^j \bmod p$ are distinct: namely, multiplication by the inverse of $g^i \bmod p$ yields $1 \bmod p$ and $g^{j-i} \bmod p$, which differ since $0 < j-i < p-1 = \text{order}(g \bmod p)$. In particular this shows that $p-1$ distinct powers of $g \bmod p$ exist. Since $(\mathbb{Z}/p\mathbb{Z})^*$ contains precisely $p-1$ elements, one concludes that every $a \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ can be written as

$$a \bmod p = g^m \bmod p,$$

for a unique m with $0 \le m < p-1$. This m is called the *discrete logarithm* of $a \bmod p$ (with respect to $g \bmod p$).

We now sketch a method to obtain some information about this discrete logarithm of $a \bmod p$ for a given primitive root $g \bmod p$. First observe that for given integers k, m one has

$$\begin{gathered} g^k \bmod p = g^m \bmod p \\ \Leftrightarrow \\ g^{|k-m|} \bmod p = 1 \bmod p \\ \Leftrightarrow \\ p-1 | k-m \\ \Leftrightarrow \\ k \equiv m \bmod (p-1). \end{gathered}$$

(Here Lemma 2.1 is used to show the middle equivalence.) So, once any k is found such that $g^k \bmod p = a \bmod p$, then the discrete logarithm of $a \bmod p$ is obtained as the remainder of the division of k by $(p-1)$.

We assume from now on that p is odd (the case $p=2$ is uninteresting). This implies that $(p-1)/2$ is a positive integer $< p-1$, hence

$$\overline{a} := g^{(p-1)/2} \bmod p$$

satisfies $\overline{a} \neq 1 \bmod p$ and $\overline{a}^2 = 1 \bmod p$. Part (4) of Lemma 2.1 then implies $\overline{a} = -1 \bmod p$. Conclusion:

$$g^{(p-1)/2} \bmod p = \overline{-1}.$$

We call $a \bmod p$ a *square modulo* p if $b \bmod p$ exists with $b^2 \bmod p = a \bmod p$.

Lemma 2.5. *$a \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ is a square modulo p if and only if $a^{(p-1)/2} = 1 \bmod p$.*

Proof. $\Rightarrow$: Suppose $a \bmod p = b^2 \bmod p$. Then using Fermat's little theorem $a^{(p-1)/2} \bmod p = b^{p-1} \bmod p = 1 \bmod p$.
$\Leftarrow$: Suppose $a^{(p-1)/2} \bmod p = \overline{1}$. Write $a \bmod p = g^m \bmod p$, then

$g^{m(p-1)/2} \bmod p = \overline{1}$. By Lemma 2.1 therefore $(p-1)|m(p-1)/2$, so an integer k exists with $(p-1)k = m(p-1)/2$, i.e., $m = 2k$. Hence $a \bmod p = g^{2k} \bmod p$, which is a square modulo p. $\square$

The criterion given here yields a very efficient test to check whether some $a \bmod p$ is a square modulo p. Namely, $a^{(p-1)/2} \bmod p$ can be calculated, by repeatedly squaring modulo p, in roughly $\log(p)$ steps. Each step here requires (approximately) $(\log(p))^2$ time units.

Conclusion: the last bit of the discrete logarithm m of $a \bmod p$ is easily determined: m is even when $a \bmod p$ is a square, otherwise m is odd. In fact this kind of information is obtained by using that the order $p-1$ of any primitive root modulo p is even. Writing $p-1 = 2^e n$ with n an integer, and $a \bmod p = g^m \bmod p$, it is even possible to determine $m \bmod 2^e$ efficiently. In the applications it is undesirable that such information is obtained so easily. To prevent it, one chooses the prime number p in such a way that

$$p - 1 = \ell \cdot k$$

with k a small integer and ℓ a large prime. Instead of a primitive root $g \bmod p$, one now takes

$$\overline{h} = h \bmod p := g^k \bmod p.$$

By construction order$(h \bmod p) = \ell$, and

$$H := \left\{\overline{h}, \overline{h}^2, \ldots, \overline{h}^{\ell-1}, \overline{h}^{\ell} = \overline{1}\right\}$$

is a subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$ consisting of precisely ℓ elements. Every $a \bmod p \in H$ can be written as $\overline{h}^m$. About such m, which is well-defined up to multiples of ℓ, it is in general much more difficult to obtain information.

2.6. **Extracting square roots modulo p.** In the subgroup H described at the end of the previous section, one has $a^\ell \equiv 1 \bmod p$ for all $a \bmod p \in H$. As a consequence, $a^{\ell+1} \equiv a \bmod p$, and since $\ell + 1$ is even, $j := (\ell+1)/2$ is a positive integer. Hence $a^j \bmod p$ is defined, and its square is $a \bmod p$. So extracting square roots is very simple in H. A similar idea is used in the so-called Tonelli-Shanks algorithm. Given an odd prime p and a square $\overline{a} = a \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ (in other words, by Lemma 2.5, $\overline{a}^{(p-1)/2} = \overline{1}$), and moreover given $\overline{b} = b \bmod p \in (\mathbb{Z}/p\mathbb{Z})^*$ which is *not* a square (so, again by Lemma 2.5, $\overline{b}^{(p-1)/2} = \overline{-1}$), this algorithm finds a square root of $\overline{a}$ in the following way:
the Tonelli-Shanks algorithm

- Define positive integers s, q (with q odd) by

$$p - 1 = q \cdot 2^s.$$

- Put $\overline{r} := \overline{a}^{(q+1)/2}$ and $\overline{t} := \overline{a}^{q}$, then
$$\overline{r}^2 = \overline{a}^{q+1} = \overline{t} \cdot \overline{a}.$$
So if it were true that $\overline{t} = \overline{1}$, then a square root of $\overline{a}$ is found, namely $\overline{r}$.

The desired equality $\overline{t} = \overline{1}$ can be rephrased as order$(\overline{t}) = 1$. For this reason we now study the order of $\overline{t}$ in the group $(\mathbb{Z}/p\mathbb{Z})^*$ more closely. Fix a primitive root $\overline{g} \in (\mathbb{Z}/p\mathbb{Z})^*$; it exists by Theorem 2.2. Since $\overline{a}$ is a square, an integer m exists such that $\overline{a} = \overline{g}^{2m}$. It follows that
$$\overline{t}^{2^{s-1}} = \overline{a}^{q2^{s-1}} = \overline{g}^{m(p-1)} = \overline{1},$$
so by Lemma 2.1 we conclude that order$(\overline{t})$ divides 2^{s-1}. Therefore
$$\text{order}(\overline{t}) = 2^i$$
with $0 \leq i \leq s-1$.

As discussed, if $i = 0$ then $\overline{t} = \overline{1}$ and we have found a square root of $\overline{a}$. We may therefore assume $i > 0$. The idea is to adjust $\overline{t}$ and $\overline{r}$ in such a way, that the identity $\overline{r}^2 = \overline{a} \cdot \overline{t}$ remains valid, and moreover order of $\overline{t}$ changes into a smaller power of 2. This will be achieved by using $\overline{b} \in (\mathbb{Z}/p\mathbb{Z})^*$, the given non-square.
- We have
$$\text{order}(\overline{b}^q) = 2^s,$$
since $(\overline{b}^q)^{2^s} = \overline{b}^{p-1} = \overline{1}$, and $(\overline{b}^q)^{2^{s-1}} = \overline{b}^{(p-1)/2} = \overline{-1}$ (here it is used that $\overline{b}$ is not a square!). Put $\overline{c} := \overline{b}^q$. Using $\overline{c}$ we will construct a *square* which has the same order as $\overline{t}$.

We know order$(\overline{t}) = 2^i$ with $1 \leq i \leq s-1$, so $s - i \geq 1$. Consider the sequence
$$\overline{c},\ \overline{c}^2,\ \overline{c}^4,\ \ldots,\ \overline{c}^{2^{s-i}}.$$
Since $\overline{c}$ has order 2^s, the order of $\overline{c}^2$ equals 2^{s-1}, and that of $\overline{c}^4$ is 2^{s-2}. Continuing like this one finds
$$\text{order}(\overline{c}^{2^{s-i}}) = 2^{s-(s-i)} = 2^i.$$
So we found an element of the same order as $\overline{t}$. Moreover, it is a square, since $s - i \geq 1$ and therefore 2^{s-i} is even.
- A square root of $\overline{e} := \overline{c}^{2^{s-i}}$ is $\overline{d} := \overline{c}^{2^{s-i-1}}$. Multiplying both sides of $\overline{r}^2 = \overline{a}\overline{t}$ by $\overline{d}^2 = \overline{e}$ one obtains
$$(\overline{r}\overline{d})^2 = \overline{a} \cdot (\overline{t}\overline{e}).$$
Claim: the order of $\overline{t}\overline{e}$ is a power of 2, strictly smaller than 2^i. This is seen as follows.

By construction both $\overline{t}$ and $\overline{e}$ have order 2^i, and $i \geq 1$. Raising $\overline{t}$ and $\overline{e}$ to the power 2^{i-1}, one obtains an element of order

2 in $(\mathbb{Z}/p\mathbb{Z})^*$. Conclusion: both $\bar{t}^{2^{i-1}}$ and $\bar{e}^{2^{i-1}}$ equal $\overline{-1}$. And therefore

$$(\bar{t}\cdot\bar{e})^{2^{i-1}} = (\overline{-1})^2 = \bar{1}.$$

Lemma 2.1 now implies that the order of $\bar{t}\cdot\bar{e}$ divides 2^{i-1}, which proves the claim.

- Repeating the steps above one finds a sequence of pairs $(\bar{r},\bar{t})$, satisfying $\bar{r}^2 = \bar{a}\cdot\bar{t}$ and with strictly decreasing powers of 2 as the order of $\bar{t}$. So after at most $s-1$ steps we have $\bar{t}=\bar{1}$, and then $\bar{r}$ is a square root of $\bar{a}$.

The Tonelli-Shanks algorithm needs a non-square $\bar{b}$ for extracting square roots of squares. Precisely half the elements of $(\mathbb{Z}/p\mathbb{Z})^*$ are non-squares (namely, the odd powers of some primitive root). So in practice a non-square is quickly found: after n times randomly selecting an element from $(\mathbb{Z}/p\mathbb{Z})^*$, the probability that all n elements are squares equals 2^{-n}.

2.7. **Diffie-Hellman key exchange.** The Diffie-Hellman key exchange is a protocol providing two parties A and B via a public channel with a common secret key.

To this end, a prime number p is used such that $p-1=\ell k$ where ℓ is a large prime, as well as $\bar{h} := g^k \bmod p$ with g a primitive root modulo p. The pair $(p,\bar{h})$ is made public. Observe that $\text{order}(\bar{h})=\ell$. The common secret key for A and B which will be constructed, is a power of $\bar{h}$ in the group $(\mathbb{Z}/p\mathbb{Z})^*$, as follows.

- As a first step, A should have a (secret) integer a, and A computes $\bar{h}^a$. Similarly B needs a secret b and determines $\bar{h}^b$. This $\bar{h}^a$, respectively $\bar{h}^b$, is now transmitted (via the public channel!) to the other party.
- Next, A computes $(\bar{h}^b)^a = \bar{h}^{ab}$, and B computes $(\bar{h}^a)^b = \bar{h}^{ab}$. This is their common secret key.

Since communication between A and B is done via a public channel, we may assume that a third party (adversary) E interested in the common secret key of A and B, also knows the pair $(p,\bar{h})$, as well as the values $\bar{h}^a$ and $\bar{h}^b$. The security of the system therefore boils down to the question: can E efficiently determine $\bar{h}^{ab}$, or at least partial information about $\bar{h}^{ab}$, given $(p,\bar{h},\bar{h}^a,\bar{h}^b)$?

A way for E to achieve this, is using discrete logarithms modulo p: first determine from $\bar{h}$ and $\bar{h}^a$ the value $a \bmod \ell$, then from $\bar{h}$ and $\bar{h}^b$ the value $b \bmod \ell$, and finally compute $\bar{h}^{ab}$ using these values. Since calculating discrete logarithms is in general difficult (i.e., no efficient algorithm for it is known), it is not expected that along these lines the common secret key of A and B can be found within a reasonable time. No alternative way is known of efficiently finding from h and $\bar{h}^a$ and $\bar{h}^b$

the value $\overline{h}^{ab}$. The security of the Diffie-Hellman key exchange protocol is based on the *assumption* that no fast way exists to do this calculation without first determining a and b. If indeed we cannot construct such a method (i.e., if we are sufficiently ignorant!), we have obtained a very secure system...

2.8. **Solving discrete logarithms.** We have seen that the security of the Diffie-Hellman key exchange protocol depends on the assumption that computing discrete logarithms is a hard problem. In the following we present the Babystep-Giantstep-Algorithm due to Shanks that solves this problem in a finite cyclic group. Suppose you are given a finite cyclic group $G = \langle g \rangle$ (e.g. $G = (\mathbb{Z}/p\mathbb{Z})^*$ for a prime p) and an element $a \in G$. The goal is to find an integer $1 \leq m < \#G$ such that $a = g^m$. The idea behind the algorithm is that for a fixed positive integer b there are unique integers q and r with $m = bq + r$, where $0 \leq r < b$. Rewriting the equation $a = g^m$ to $a = g^m = g^{bq}g^r$ and multiplying both sides by g^{-bq}, gives us that finding m is the same as finding a pair (q, r) such that $ag^{-bq} = g^r$ holds.

In the first part of the algorithm, the *babysteps*, the elements g^k for $0 \leq k, b$ are computed. In the second part, the *giantsteps*, the elements ag^{-bq} for $q = 1, 2, \ldots$ are computed until one of the giantsteps is equal to one of the babysteps. Such a collision is granted by the considerations above.

We now discuss what is a good choice for the integer b. The first observation is that for small b only a few babysteps are needed, while a lot of giantsteps have to be computed. For large b it is the other way around. The worst case scenario is given in the case when we have to compute $\frac{\#G}{b}$ giantsteps. Thus, in the worst case there are $\frac{\#G}{b} + b$ computations in G needed. This number attains a minimum for $b \approx \sqrt{\#G}$.

2.9. **Rivest-Shamir-Adleman.** The celebrated RSA (Rivest-Shamir-Adleman) public key cryptosystem has, as the name suggests, as an important feature that the key used for encryption, is public. Hence this is totally different from, e.g., the AES, where it is essential to keep the key secret. The advantage of a public key is obviously, that it is not necessary to agree on a common shared key (for example using Diffie-Hellman key exchange).

The safety of RSA relies on the assumption, that for general N and e no efficient algorithm exists for extracting e-th roots of integers modulo N. Taking here $N = p$ prime and $e = 2$, we know from (2.6) above that such an algorithm *does* exist. Even simpler, again for $N = p$ prime, if one takes $e > 0$ such that $\gcd(e, p-1) = 1$, then an e-th root of $\overline{a} \in \mathbb{Z}/p\mathbb{Z}$ is easily constructed as follows: compute using the extended Euclidean algorithm $d > 0$ with $ed \equiv 1 \bmod (p-1)$. Then $\overline{a}^d$ is the desired root, because if $\overline{a} = \overline{0}$ then clearly $\overline{0}^d = \overline{0}$ works, and is $\overline{a} \neq \overline{0}$, then because p is prime, $\overline{a} \in (\mathbb{Z}/p\mathbb{Z})^*$ and

$$(\overline{a}^d)^e = \overline{a} \cdot \overline{a}^{ed-1} = \overline{a} \cdot \overline{1} = \overline{a},$$

since $ed - 1$ is a multiple of $p - 1$. So we found an e-th root of $\overline{a}$.

RSA does not use a prime number N, but instead takes N a product of two large distinct primes. The system works as follows.

- Take two large primes $p \neq q$ and calculate $N = pq$. The integer N is made public, but p and q are kept secret.
- One has $\varphi(N) = (p-1)(q-1)$ as we saw earlier in this chapter. The value $\varphi(N)$ is also kept secret. Next, take $e > 1$ such that $\gcd(e, \varphi(N)) = 1$. This e is the public key.
- If anybody wants to send us a message $\overline{m} \in \mathbb{Z}/N\mathbb{Z}$, he/she computes $\overline{m}^e$. This is the encrypted version of the message, which is now sent to us.
- We know $\varphi(N)$ as well as e, and e was chosen a unit modulo $\varphi(N)$. We can therefore compute $d > 0$ such that $de \equiv 1 \bmod \varphi(N)$. Then $(\overline{m}^e)^d$ equals the original message $\overline{m}$.

To see why this system works, we need to verify that $\overline{m}^{de} = \overline{m}$ for every $\overline{m}$. In other words: all $m \in \mathbb{Z}$ have the property that N divides $m^{de} - m$. Since $N = pq$ with p and q distinct primes, this is equivalent to the assertion that both p and q divide $m^{de} - m$. Since $de - 1$ is a multiple of $\varphi(N) = (p-1)(q-1)$, it is a multiple of $p - 1$ and $q - 1$, too. The same argument used above for extracting e-th roots modulo p now finishes the reasoning.

It is clear why in this system $\varphi(N)$ needs to be secret: e is public, so knowing $\varphi(N)$ it is easy to find d with $de \equiv 1 \bmod \varphi(N)$, and this compromises the system. If one knows the divisors p, q of N, one also knows $\varphi(N)$. Conversely, knowing $\varphi(N)$ implies knowing the sum $p+q$ which is $s := N + 1 - \varphi(N)$. Since $N = pq$, now p and q are the two solutions of the equation

$$x^2 - sx + N = 0,$$

which are easy to find. Factoring N is therefore equally difficult as determining $\varphi(N)$. Since we cannot do this efficiently for very large N, this is not seen as a danger for the RSA system (NIST advises, at present, to take p, q so large that N is roughly of size 2^{2048}).

One could imagine that some efficient algorithm exists for extracting e-th roots modulo N, without using $\varphi(N)$ or the prime factorisation of N. Such an algorithm would make RSA unsafe. However, nobody appears to have any idea how such an algorithm should look like, so one assumes RSA to be secure. Of course then $N = pq$ should be difficult to factor. For this reason one avoids primes p such that $p + m$, for some integer m with $|m| < 2\sqrt{p}$, factors into many small primes.

As an example primes p with $p+1 = 2^n$ (these are called Mersenne primes) are unfit for RSA.

Although the description given here presents the basic idea of RSA, in practice additional issues are necessary. For example: if the encryption key is public, then without further knowledge it will be hard to verify that a received message was actually sent by the person asserting he/she sent it. Solutions for this kind of problems exist, and one of them is discussed in Section 2.10.

2.10. **ElGamal digital signatures.** Taher ElGamal obtained his PhD in 1984 supervised by Martin Hellman, one of the two persons we encountered describing the Diffie-Hellman protocol. He presented a method to equip messages m with a digital signature. It works as follows.

The messages m to be sent, are taken from some set M. This could be $\mathbb{Z}/N\mathbb{Z}$ (as in the case of RSA), or the set of 'states' when using AES, or (as with Diffie-Hellman) a group $(\mathbb{Z}/p\mathbb{Z})^*$. In the ElGamal system a *hash function* plays a role; this is a function

$$H \ : \ M \longrightarrow \mathbb{Z}_{>0}.$$

Hash functions are often used in cryptography. For various practical applications hash functions have been constructed, with fancy names such as SHA-1 and SHA-2 and SHA-3 (SHA = Secure Hash Algorithm). SHA-3 was released by NIST quite recently: August 5th, 2015, see `http://www.nist.gov/itl/csd/201508_sha3.cfm`. A condition any hash function needs to satisfy, is that no efficient way is known which, on input some value $H(m)$, outputs $\tilde{m} \in M$ such that $H(\tilde{m}) = H(m)$. We will assume that the function H is public; given $m \in M$, anyone can calculate the value $H(m)$.

Next, a large prime p and a primitive root $\overline{g} \in (\mathbb{Z}/p\mathbb{Z})^*$ are chosen. The person who wants to add a digital signature to the message m, owns (or chooses) a secret key x: it is an integer satisfying $1 < x < p-1$. He/she now calculates $\overline{y} := \overline{g}^x$. The triple

$$(p, \overline{g}, \overline{y})$$

is called the public key (of this person). This public key, as the name suggests, is made public.

A digital signature on the message m is now constructed as follows.

- Pick a random integer k such that $k \bmod (p-1)$ is a unit;
- Compute r with $1 \leq r \leq p-1$ such that $r \bmod p = \overline{g}^k$;
- Compute s with $0 \leq s < p-1$ such that

$$s \bmod (p-1) = ((H(m) - xr) \bmod (p-1)) \cdot (k \bmod (p-1))^{-1}.$$

- If $s = 0$, take a different random k, until $s \neq 0$.

The ElGamal digital signature on m is the pair (r, s).

The only way in which in this recipe $s = 0$ can occur, is that r satisfies $xr \equiv H(m) \bmod (p-1)$. Now x is fixed, and satisfies $x \not\equiv 0 \bmod (p-1)$. As a consequence, multiplication by x as a map $\mathbb{Z}/(p-1)\mathbb{Z} \to \mathbb{Z}/(p-1)\mathbb{Z}$ is not a constant map. In particular, $H(m) \bmod (p-1)$ can not be the only element of the image of this map (it is possible that it is not in the image at all!). This shows that a value r exists such that the corresponding $s \neq 0$. Note that in the algorithm constructing a digital signature, only integers r satisfying $r \bmod p = \overline{g}^k$ with $\gcd(k, p-1) = 1$, are considered. These are exactly all primitive roots in $(\mathbb{Z}/p\mathbb{Z})^*$:

Lemma 2.11. *Let p be an odd prime and $\overline{g} \in (\mathbb{Z}/p\mathbb{Z})^*$ a primitive root. For integers k one has*

$$\mathrm{order}(\overline{g}^k) = p - 1 \Leftrightarrow \gcd(k, p-1) = 1.$$

Consequently, there exist precisely $\varphi(p-1)$ primitive roots modulo p.

Proof. Put $d := \mathrm{order}(\overline{g}^k)$. Then $(p-1)|dk$, so if $\gcd(k, p-1) = 1$ then it follows that $(p-1)|d$. By Lemma 2.1 $d|(p-1)$, hence $d = p-1$.

In case $\gcd(k, p-1) = e > 1$, put $k = ek_1$ and $p - 1 = ee_1$. The equality

$$(\overline{g}^k)^{e_1} = \overline{g}^{k_1 e e_1} = (\overline{g}^{p-1})^{k_1} = \overline{1}$$

now shows that $d = \mathrm{order}(\overline{g}^k) \leq e_1 < p - 1$.

Since $\overline{g}^k$ only depends on $k \bmod (p-1)$, the above argument shows that the primitive roots modulo p are in a one to one correspondence with the $k \bmod (p-1) \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$. There exist $\varphi(p-1)$ of those. □

We have that $\varphi(p-1)$ is large when p is a large prime, so the condition that $s \neq 0$, is in general not a problem. For example, one could put as an additional condition that the secret key x has to be a unit modulo $(p-1)$. This additional condition implies that a unique $r \bmod (p-1)$ exists with $xr \equiv H(m) \bmod (p-1)$. In case $\varphi(p-1) > 1$ (which holds for primes $p > 7$), the added condition guarantees we will find an $s \neq 0$. In practice none of this is a serious problem.

If a receiver obtains a message m equipped with a digital signature (r, s), he/she calculates $H(m)$ and $\overline{g}^{H(m)}$ (this is possible because $\overline{g}$ and the Hash function H are public). The sender also made a public key $\overline{y}$; the receiver now computes

$$\overline{y}^r \cdot (r^s \bmod p)$$

and verifies that this equals $\overline{g}^{H(m)}$.

Since the digital signature needs to satisfy

$$ks \equiv (H(m) - xr) \bmod (p-1),$$

one has indeed

$$\overline{g}^{H(m)} = \overline{g}^{ks}\overline{g}^{xr} = (r^s \bmod p) \cdot \overline{y}^r,$$

hence if the verification by the receiver did not result in equality, then either the message or the signature had been tampered with.

The usefulness and security of this system rely on two assumptions:

(1) It is not easy to find the secret key of the sender. In other words, given the public $\overline{y} = \overline{g}^x$ it is not easy to retrieve x.
(2) It is difficult to falsify a digital signature. i.e., even with $\overline{y}$ and $\overline{g}$ being public, it is not easy to construct a pair (r, s) (without using the secret key x) which will be regarded as a valid signature on a message $\tilde{m}$. It may sound plausible that this indeed holds, yet no formal argument is known why such an efficient construction can not exist.

3. Prime numbers

The given applications in cryptography require that we have a sufficient supply of large primes at our disposal. Here we discuss a practical algorithm to find such primes: the Miller-Rabin test. It is based on the fact that the equation

$$x^2 - \overline{1} = \overline{0}$$

has only two solutions in $\mathbb{Z}/p\mathbb{Z}$ (for p an odd prime), namely $x = \overline{1}$ and $x = \overline{-1}$. A similar assertion was used in the proof of Theorem 2.2. A short proof: let $n \bmod p$ be a solution. Then $p|(n^2-1) = (n-1)(n+1)$. Since p is prime, it follows that $p|n-1$ or $p|n+1$, in other words, $n \equiv \pm 1 \bmod p$. (Alternative proof: by Lemma 2.1 $(\mathbb{Z}/p\mathbb{Z})^*$ contains at most, and hence exactly, 2 elements of order dividing 2.)

Let p be an odd prime and write, as when discussing the Tonelli-Shanks algorithm, $p - 1 = q2^s$ with $s > 0$ and q odd. For an integer a such that $1 \le a \le p-1$ we have that order$(\overline{a}^q)$ divides 2^s, so this order equals 2^i with $0 \le i \le s$. If $s = 0$, then $\overline{a}^q = \overline{1}$. If $s > 0$, then $\overline{a}^{q2^{i-1}}$ has order 2, hence $\overline{a}^{q2^{i-1}} = \overline{-1}$. In other words: the sequence

$$\overline{a}^q,\ \overline{a}^{q2}, \ldots, \overline{a}^{q2^{s-1}}$$

contains the element $\overline{-1}$.

This observation leads to a simple test: suppose we have an odd integer $n \in \mathbb{Z}_{\ge 3}$ and we want to test whether it is prime. The Miller-Rabin test attempts this as follows.

- Put $n - 1 = q2^s$ with q odd and $s > 0$. Repeat the following steps a number of times:
- Randomly choose $\overline{a} \ne \overline{0}$ in $\mathbb{Z}/n\mathbb{Z}$ and compute $\overline{b} := \overline{a}^q$.
- If $\overline{b} = \overline{1}$, try another $\overline{a}$.

- If $\overline{b} \neq \overline{1}$, compute the sequence
$$\overline{b},\ \overline{b}^2, \ldots, \overline{b}^{2^{s-1}}$$
and check if it contains $\overline{-1}$. If not, then n is composite and the algorithm terminates. Otherwise, try another $\overline{a}$.
- If the test is passed for sufficiently many $\overline{a}$'s, then n is probably prime.

In case this algorithm outputs that n is composite, then indeed it is. Namely, it means that $\overline{a} \neq \overline{0}$ in $\mathbb{Z}/n\mathbb{Z}$ was found, with properties different from any element in $(\mathbb{Z}/p\mathbb{Z})^*$ for p prime, as we saw earlier.

If the algorithm outputs that n is probably prime, it means that *every* $\overline{a}$ considered, satisfies either $\overline{a}^q = \overline{1}$ (in that case $\overline{a}$ is a unit modulo n, namely one of order dividing q), or $\overline{a}^{q2^i} = \overline{-1}$ for an i with $0 \leq i < s$. In this case $\overline{a}$ is also a unit modulo n, now with even order. In short, the conclusion is that all used $\overline{a}$'s are in
$$A := \left\{ \overline{a} \in (\mathbb{Z}/n\mathbb{Z})^* \mid \overline{a}^q = \overline{1} \ \text{ or } \ \exists 0 \leq i < s :\ \overline{a}^{q2^i} = \overline{-1} \right\}.$$
This set A satisfies:

Theorem 3.1. *If the odd integer $n > 1$ is composite, then*
$$\frac{\#A}{\varphi(n)} \leq \frac{1}{4}.$$

A proof of this is presented in a text by René Schoof: `http://http://www.mat.uniroma2.it/~schoof/05rene.pdf`. It uses a result we now prove using Theorem 2.2:

Lemma 3.2. *Let p be an odd prime and $e > 0$. Then $\varphi(p^e) = p^{e-1}(p-1)$, and $(\mathbb{Z}/p^e\mathbb{Z})^*$ contains an element of order $p^{e-1}(p-1)$.*

Proof. By definition $\varphi(p^e)$ equals the number of integers n such that $1 \leq n \leq p^e$ and $\gcd(n, p^e) = 1$, which means p does not divide n. So one finds p^e minus the number of $n \in \{1, \ldots, p^e\}$ that *are* multiples of p. Hence $\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1)$.

The application
$$a + p^e\mathbb{Z} \mapsto a + p\mathbb{Z}$$
defines a map
$$f :\ (\mathbb{Z}/p^e\mathbb{Z})^* \longrightarrow (\mathbb{Z}/p\mathbb{Z})^*$$
satisfying $f(\overline{a} \cdot \overline{b}) = f(\overline{a})f(\overline{b})$ for units $\overline{a}, \overline{b}$. This map is surjective, as an arbitrary $a + p\mathbb{Z}$ has preimage (e.g.) $a + p^e\mathbb{Z}$. Take a primitive root $g + p\mathbb{Z} \in (\mathbb{Z}/p\mathbb{Z})^*$ and put $\overline{g} := g + p^e\mathbb{Z}$. Write $d := \text{order}(\overline{g})$. Since $\overline{g}^d = \overline{1}$, also $g^d \bmod p = f(\overline{g}^d) = f(\overline{1}) = 1 \bmod p$. Lemma 2.1 implies that d is a multiple of the order of $g \bmod p$, which is $p-1$. Therefore $d = (p-1)d_1$ for some $d_1 \geq 1$.

Then the element $\overline{h} := \overline{g}^{d_1}$ has order $p-1$ in the group $(\mathbb{Z}/p^e\mathbb{Z})^*$. If we moreover find an element $\overline{j} \in (\mathbb{Z}/p^e\mathbb{Z})^*$ with $\text{order}(\overline{j}) = p^{e-1}$, then part (3) of Lemma 2.1, observing that $\gcd(p^{e-1}, p-1) = 1$, shows that $\overline{hj}$ is the desired element of order $\varphi(p^e)$ is.

Claim: $\overline{j} := (1+p) \bmod p^e$ is in $(\mathbb{Z}/p^e\mathbb{Z})^*$, and its order equals p^{e-1}. Namely, from the binomial formula (and induction on e) one finds $(1+p)^{p^{e-1}} \equiv 1 \bmod p^e$. Therefore $(1+p) \bmod p^e$ is a unit, and its order divides p^{e-1}. Write this order as p^i with i satisfying $0 \leq i \leq e-1$. Then

$$p^e | \left((1+p)^{p^i} - 1 \right).$$

For contradiction, if $i < e-1$, then also $i+2 \leq e$, and therefore

$$(1+p)^{p^i} \equiv 1 \bmod p^{i+2}.$$

However, again from the binomial formula and induction to i, one has

$$(1+p)^{p^i} \equiv (1+p^{i+1}) \bmod p^{i+2}.$$

Combining the congruences yields $p^{i+1} \equiv 0 \bmod p^{i+2}$, a contradiction! So indeed $\overline{j}$ has order p^{e-1}, completing the proof. □

4. A factorisation method: Pollard $p-1$

Discussing RSA, we saw that a necessary condition for its security is, that no efficient algorithm is known which, on input of a composite integer n, outputs a divisor $m|n$ with $1 < m < n$. This leads to the question, which algorithms have been found so far for this problem. The most successful algorithms to date (2015) are the "number field sieve" (two versions of it, the 'special' SNF, suited for special integers such as $2^{2^m}+1$ and many more, and the 'general' GNF, used for numbers with no specific additional form. A predecessor of the number field sieve, the so-called quadratic sieve, factored several large numbers as well. An other famous algorithm is the "elliptic curve method", ECM. Especially when close to a prime divisor p of n, many numbers $p+a$ occur which have only small prime factors, then ECM, a method proposed by Leiden mathematician H.W. Lenstra (we met him when discussing AES), turns out to be very efficient.

In this section we describe an algorithm which provided an important motivation for ECM. It was proposed by the English mathematician John Pollard, and it is called the $p-1$ method. The most important difference between Pollard $p-1$ and ECM, is that the first uses the group $(\mathbb{Z}/p\mathbb{Z})^*$ for some prime $p|n$, whereas ECM instead uses the so-called group of points over $\mathbb{Z}/p\mathbb{Z}$ of an elliptic curve. We will discuss elliptic curves in Section 5. The idea in Pollard $p-1$ is as follows.

Suppose $n > 1$ is composite. If p is a prime divisor of n, then we obtain a surjective map

$$\pi: \ (\mathbb{Z}/n\mathbb{Z})^* \longrightarrow (\mathbb{Z}/p\mathbb{Z})^*$$

given as $\overline{a} = a + n\mathbb{Z} \mapsto a + p\mathbb{Z}$. The equality $\pi(\overline{a} \cdot \overline{b}) = \pi(\overline{a})\pi(\overline{b})$ holds. If e is some divisor of $p-1$, then $a + p\mathbb{Z}$ in $(\mathbb{Z}/p\mathbb{Z})^*$ exists satisfying $a^e \bmod p = 1 \bmod p$, and therefore, for any multiple E of e, also $a^E \bmod p = 1 \bmod p$. Elements $a \bmod p$ as discussed here, are easy to describe: if $g \bmod p$ is a primitive root modulo p, writing $p-1 = de$, then

$$g^d \bmod p,\ g^{2d} \bmod p, \ldots, g^{d(e-1)} \bmod p,\ 1 \bmod p$$

are the desired classes $a \bmod p$.

Randomly picking a unit $a \bmod p$, the probability that it is one of the classes above, equals $e/(p-1) = 1/d$. We can do the same for other divisors e of $p-1$. Then, provided $p-1$ has many divisors, the probability that $a^E \bmod p = 1 \bmod p$, with E a common multiple of several such e's, is not too small. Of course we do not know the prime divisor p, and hence neither the suitable e's. What can be done however, is guess a possible E. It should have many divisors in order to increase the probability that $a^E \equiv 1 \bmod p$, or equivalently, that p divides $a^E - 1$. A reasonable choice is to take

$$E := \mathrm{lcm}(2, 3, 4, 5, \ldots, B)$$

for a not too small bound B. In Magma this is implemented as follows:

```
E:=1; B:=100;
for j in [2..B] do
  E:=LCM(E,j);
end for;
E;
```

Now randomly pick a (small) integer a, for example, a small prime. We do not know the desired prime factor p of n (yet), and thus neither do we know $a^E \bmod p$. But we *can* compute $a^E \bmod n$. Write $a^E \bmod n = b \bmod n$. Our hope is that $a^E \bmod p = 1 \bmod p$, i.e., $p|b-1$. If this were true, then p divides n as well as $b-1$, so $p|\gcd(n, b-1)$. This is precisely the way in which the Pollard $p-1$ algorithm attempts to find divisors of n. With E as above, it is implemented in Magma as follows:

```
pollard := function(n)
        R := quo<Integers()|n>;
        a := 2;
        for j in [1..100] do
                a := NextPrime(a);
                if (n mod a) eq 0 then
                    return a;
                else
                    b := (R!a)^E;
                    g := GCD(n,b-1);
                    if not (g eq 1) and not (g eq n) then
                        return g;
                    end if;
```

```
            end if;
        end for;
        return 0;
    end function;
pollard(10511111111111);
pollard(2*3^40+1);
```

5. Elliptic curves

The abbreviation ECC in cryptography stands for "Elliptic Curve Cryptography": cryptography using elliptic curves. Much has been written about this subject. A first and incomplete description is, that the role of $(\mathbb{Z}/p\mathbb{Z})^*$ in various protocols, may be replaced by the group $E(\mathbb{Z}/p\mathbb{Z})$ of points with coordinates in the integers modulo p, on an elliptic curve E.

Here we present a brief introduction to the theory of elliptic curves. We closely follow a part of the slides made by the American mathematician Joe Silverman in 2006 for a summer school in Wyoming on the subject. In his text he writes, as is common in algebra, $\mathbb{F}_p$ for the field $\mathbb{Z}/p\mathbb{Z}$. If $q = p^e$ is a power of a prime p, then more generally $\mathbb{F}_q$ denotes a field consisting of precisely q elements. For example, when discussing AES we encountered $\mathbb{F}_{256}$.

The complete text by Silverman can be found at `www.math.brown.edu/~jhs/Presentations/WyomingEllipticCurve.pdf`; a small part of this text is found below, with some additional explanation.

An Introduction to the Theory of Elliptic Curves

Joseph H. Silverman

Brown University and
NTRU Cryptosystems, Inc.

Summer School on
Computational Number Theory and Applications to Cryptography
University of Wyoming
June 19 – July 7, 2006

0

What is an Elliptic Curve?

- An elliptic curve is a curve that's also naturally a group.
- The group law is constructed geometrically.
- Elliptic curves have (almost) nothing to do with ellipses, so put ellipses and conic sections out of your thoughts.
- Elliptic curves appear in many diverse areas of mathematics, ranging from number theory to complex analysis, and from cryptography to mathematical physics.

Points on Elliptic Curves

- Elliptic curves can have points with coordinates in any field, such as $\mathbb{F}_p$, $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$.
- Elliptic curves with points in $\mathbb{F}_p$ are finite groups.
- **Elliptic Curve Discrete Logarithm Problem** (**ECDLP**) is the discrete logarithm problem for the group of points on an elliptic curve over a finite field.
- The best known algorithm to solve the ECDLP is exponential, which is why elliptic curve groups are used for cryptography.
- More precisely, the best known way to solve ECDLP for an elliptic curve over $\mathbb{F}_p$ takes time $O\big(\sqrt{p}\big)$.
- The goal of these talks is to tell you something about the theory of elliptic curves, with an emphasis on those aspects that are of interest in cryptography.

The Equation of an Elliptic Curve

An **Elliptic Curve** is a curve given by an equation of the form

$$\boxed{y^2 = x^3 + Ax + B}$$

There is also a requirement that the **discriminant**

$$\Delta = 4A^3 + 27B^2 \text{ is nonzero.}$$

Equivalently, the polynomial $x^3 + Ax + B$ has distinct roots. This ensures that the curve is nonsingular.

For reasons to be explained later, we also toss in an extra point, $\mathcal{O}$, that is "at infinity," so E is the set

$$E = \big\{(x, y) : y^2 = x^3 + Ax + B\big\} \cup \{\mathcal{O}\}.$$

Amazing Fact: We can use **geometry** to make the points of an elliptic curve into a group. The next few slides illustrate how this is accomplished.

Explanation: let $s = (\alpha, \beta)$ be a point on E, so $\beta^2 = \alpha^3 + A\alpha + B$. A line containing s can be given as

$$\ell = \{s + tr\}$$

with $r \neq (0, 0)$ a fixed direction and t a parameter. Write

$$F(x, y) = x^3 + Ax + B - y^2.$$

The intersection of ℓ and E corresponds to the values t such that $F(s+tr) = 0$. The expression $F(s+tr)$ is a polynomial in the variable t, and its degree is at most 3. Moreover, $t = 0$ is a zero of this polynomial, since $s \in E$.

We call ℓ a tangent line to E in s, if the zero $t = 0$ of $F(s + tr)$ has multiplicity at least 2. Equivalently, if the polynomial $F(s + tr)$ is divisible by t^2. Note that this definition of "tangent line" in the case that we work over the real numbers, agrees with our intuition for tangency. However, the definition does not only make sense over $\mathbb{R}$: it works equally well over an arbitrary field. To find such a tangent line, the direction $r = (\gamma, \delta)$ should be chosen in such a way that $t = 0$ is a double zero of $F(s + tr)$. Since the coefficient of t in this polynomial equals

$$\gamma\frac{\partial F}{\partial x}(s) + \delta\frac{\partial F}{\partial y}(s) = \gamma(3\alpha^2 + A) - 2\delta\beta,$$

we observe that in general, in a given point $s = (\alpha, \beta)$ of E one finds a unique tangent line ℓ, namely the line with as direction r an arbitrary multiple of $(2\beta, 3\alpha^2 + A)$.

A point s is called a *singular point* of the curve, if it is on the curve and moreover there is more than one tangent line in s to the curve. The above calculation shows that $s = (\alpha, \beta)$ is a singular point of E, if it satisfies besides the equation of E also the system

$$\begin{cases} 2\beta = 0 \\ 3\alpha^2 + A = 0. \end{cases}$$

If $2 \neq 0$ holds in the field we consider, then a singular point on E is necessarily of the form $s = (\alpha, 0)$, with α a zero of $x^3 + Ax + B$ of multiplicity at least 2. In particular this implies

$$A = -3\alpha^2$$

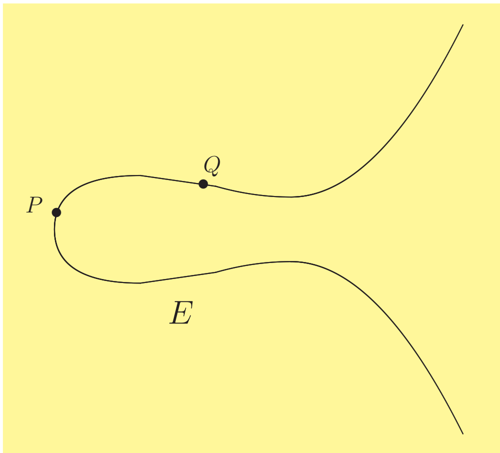
and

$$B = -\alpha^3 - A\alpha = -\alpha^3 + 3\alpha^3 = 2\alpha^3.$$

Combining the two equalities above, we get $4A^3 + 27B^2 = 0$. So if $4A^3 + 27B^2 \neq 0$, then the curve contains no singular points, i.e., in every point on the curve one has a unique tangent line.

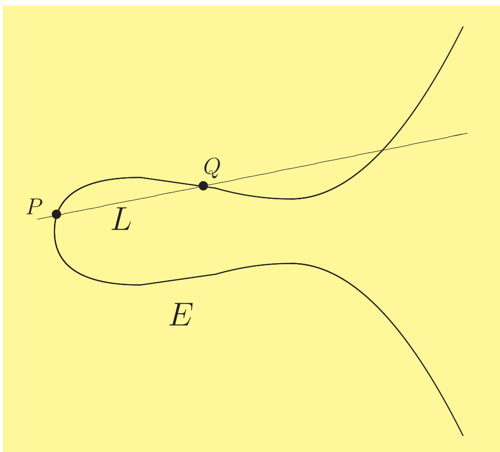
This tangent line assumption is used throughout Silverman's slides.

Adding Points on an Elliptic Curve



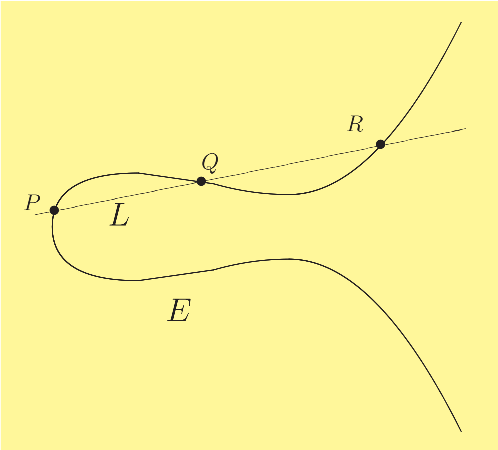
Start with two points P and Q on E.

Adding Points on an Elliptic Curve



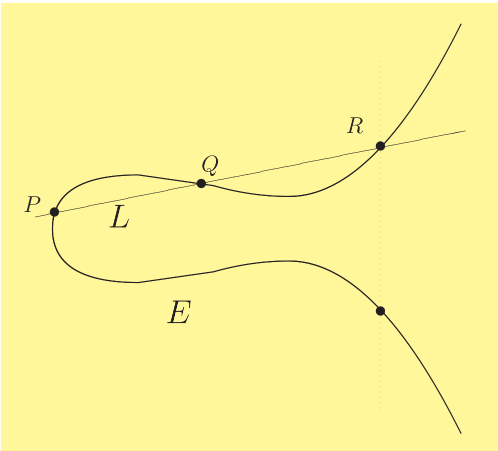
Draw the line L through P and Q.

Adding Points on an Elliptic Curve



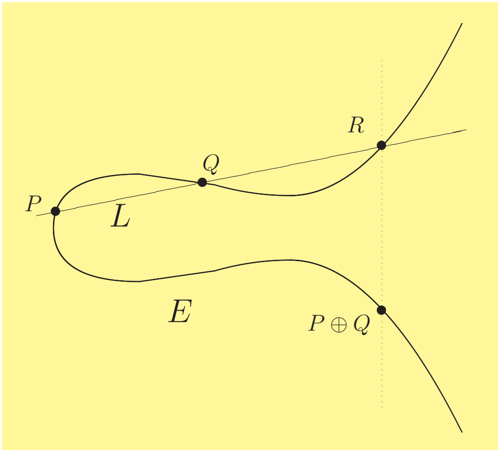
The line L intersects the cubic curve E in a third point. Call that third point R.

Adding Points on an Elliptic Curve



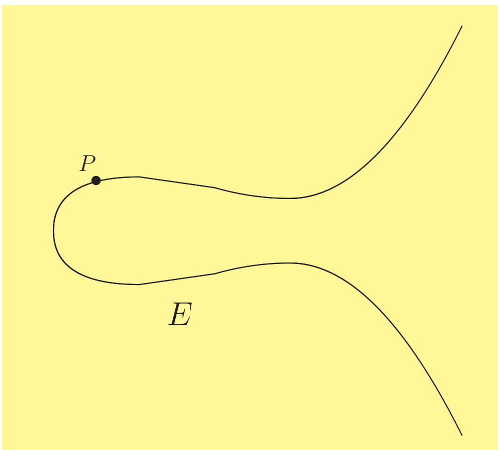
Draw the vertical line through R.
It hits E in another point.

Adding Points on an Elliptic Curve



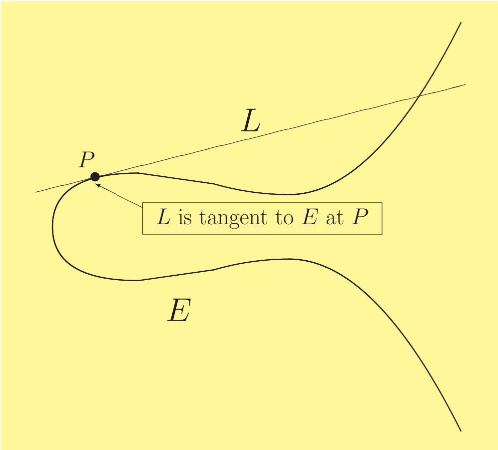
We define the **sum of P and Q on E** to be the reflected point. We denote it by $P \oplus Q$ or just $P + Q$.

Adding a Point To Itself on an Elliptic Curve



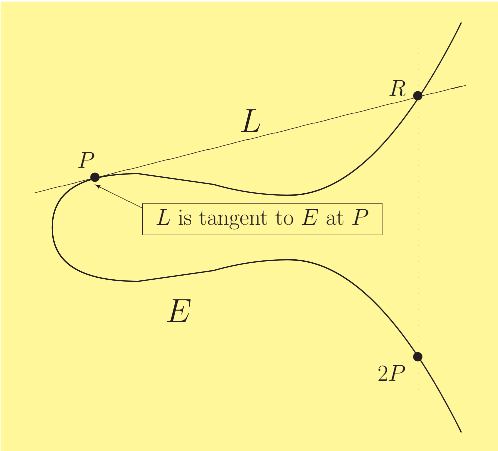
How do we add a point P to itself, since there are many different lines that go through P?

Adding a Point To Itself on an Elliptic Curve



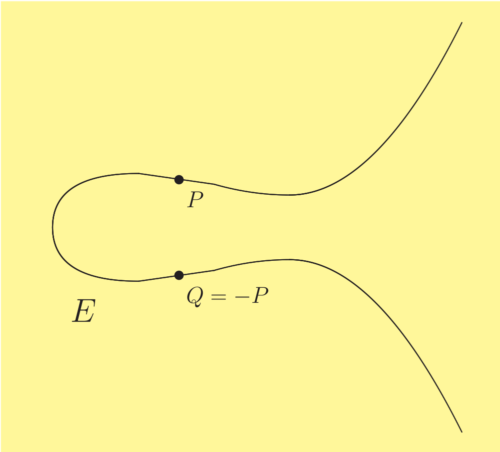
If we think of adding P to Q and let Q approach P, then the line L becomes the tangent line to E at P.

Adding a Point To Itself on an Elliptic Curve



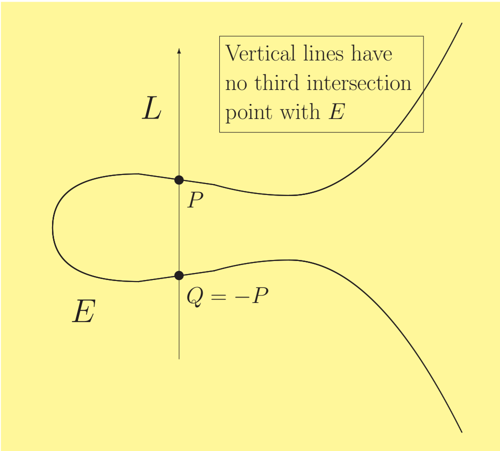
Then we take the third intersection point R, reflect across the x-axis, and call the resulting point $P \oplus P$ or $2P$.

Vertical Lines and the Extra Point "At Infinity"



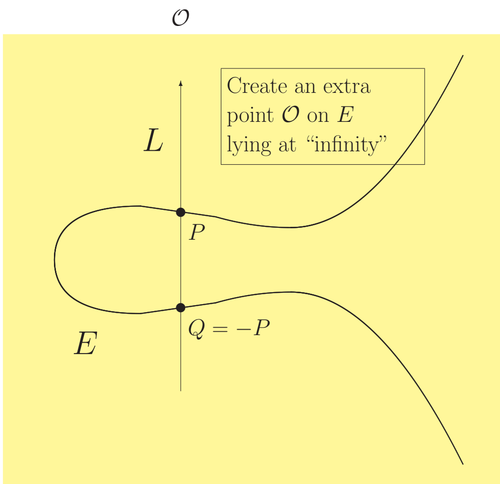
Let $P \in E$. We denote the reflected point by $-P$.

Vertical Lines and the Extra Point "At Infinity"



Big Problem: The vertical line L through P and $-P$ does not intersect E in a third point! And we need a third point to define $P \oplus (-P)$.

Vertical Lines and the Extra Point "At Infinity"



Solution: Since there is no point in the plane that works, we create an extra point $\mathcal{O}$ "at infinity."

Rule: $\mathcal{O}$ is a point on every vertical line.

Properties of "Addition" on E

Theorem *The addition law on E has the following properties:*

(a) $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E$.
(b) $P + (-P) = \mathcal{O}$ for all $P \in E$.
(c) $P + (Q + R) = (P + Q) + R$ for all $P, Q, R \in E$.
(d) $P + Q = Q + P$ for all $P, Q \in E$.

In other words, the addition law $+$ makes the points of E into a commutative group.

All of the group properties are trivial to check **except** for the associative law (c). The associative law can be verified by a lengthy computation using explicit formulas, or by using more advanced algebraic or analytic methods.

Adding points on an elliptic curve can (for example) be done using the Magma package. The example presented below considers the curve with equation $y^2 = x^3+2x+9$ over the field of rational numbers $\mathbb{Q}$. Two points on this curve are $P := (0, 3)$ and $P_2 := (4, 9)$. Combinations of these points are found as follows.

```
Q:=Rationals();
E:=EllipticCurve([ Q | 2, 9]);E;
P:=E![0,3];P2:=E![4,9];
2*P;
P-P2;
6*P+7*P2;
```

The geometrically described rules for adding points on an elliptic curve will now be described with formulas.

Formulas for Addition on E

Suppose that we want to add the points

$$P_1 = (x_1, y_1) \quad \text{and} \quad P_2 = (x_2, y_2)$$

on the elliptic curve

$$E : y^2 = x^3 + Ax + B.$$

Let the line connecting P to Q be

$$L : y = \lambda x + \nu$$

Explicitly, the slope and y-intercept of L are given by

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \dfrac{3x_1^2 + A}{2y_1} & \text{if } P_1 = P_2 \end{cases} \qquad \text{and} \qquad \nu = y_1 - \lambda x_1.$$

Formulas for Addition on E (continued)

We find the intersection of

$$E : y^2 = x^3 + Ax + B \qquad \text{and} \qquad L : y = \lambda x + \nu$$

by solving

$$(\lambda x + \nu)^2 = x^3 + Ax + B.$$

We already know that x_1 and x_2 are solutions, so we can find the third solution x_3 by comparing the two sides of

$$\begin{aligned} &x^3 + Ax + B - (\lambda x + \nu)^2 \\ &\quad = (x - x_1)(x - x_2)(x - x_3) \\ &\quad = x^3 - (x_1 + x_2 + x_3)x^2 + (x_1x_2 + x_1x_3 + x_2x_3)x - x_1x_2x_3. \end{aligned}$$

Equating the coefficients of x^2, for example, gives $-\lambda^2 = -x_1 - x_2 - x_3$, and hence $x_3 = \lambda^2 - x_1 - x_2$. Then we compute y_3 using $y_3 = \lambda x_3 + \nu$, and finally

$$P_1 + P_2 = (x_3, -y_3).$$

Formulas for Addition on E (Summary)

Addition algorithm for $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on the elliptic curve $E : y^2 = x^3 + Ax + B$

- If $P_1 \neq P_2$ and $x_1 = x_2$, then $P_1 + P_2 = \mathcal{O}$.
- If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = 2P_1 = \mathcal{O}$.
- If $P_1 \neq P_2$ (and $x_1 \neq x_2$),

$$\text{let } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ and } \nu = \frac{y_1x_2 - y_2x_1}{x_2 - x_1}.$$

- If $P_1 = P_2$ (and $y_1 \neq 0$),

$$\text{let } \lambda = \frac{3x_1^2 + A}{2y_1} \text{ and } \nu = \frac{-x^3 + Ax + 2B}{2y}.$$

Then

$$P_1 + P_2 = (\lambda^2 - x_1 - x_2, -\lambda^3 + \lambda(x_1 + x_2) - \nu).$$

An Observation About the Addition Formulas

The addition formulas look complicated, but for example, if $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are distinct points, then

$$x(P_1 + P_2) = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2,$$

and if $P = (x, y)$ is any point, then

$$x(2P) = \frac{x^4 - 2Ax^2 - 8Bx + A^2}{4(x^3 + Ax + B)}.$$

Important Observation: If A and B are in a field K and if P_1 and P_2 have coordinates in K, then $P_1 + P_2$ and $2P_1$ also have coordinates in K.

The Group of Points on E with Coordinates in a Field K

The elementary observation on the previous slide leads to the important result that points with coordinates in a particular field form a subgroup of the full set of points.

Theorem. (Poincaré, ≈ 1900) Let K be a field and suppose that an elliptic curve E is given by an equation of the form

$$E : y^2 = x^3 + Ax + B \qquad \text{with} \quad A, B \in K.$$

Let $E(K)$ denote the set of points of E with coordinates in K,

$$E(K) = \big\{(x, y) \in E : x, y \in K\big\} \cup \{\mathcal{O}\}.$$

Then $E(K)$ is a **subgroup** of the group of all points of E.

With Magma it is easy to compute in a group $E(\mathbb{Z}/p\mathbb{Z})$ for p a prime number:

```
p:=37;
Fp:=GF(p);
E:=EllipticCurve([ Fp | -5,8]);
P:=E![6,3]; Q:=E![10,12];
Order(P);
Order(Q);
RationalPoints(E);
#E;
```

In the given example all 45 points in $E(\mathbb{F}_{37})$ turn out to be combinations of P and Q.

A Finite Field Example (continued)

Substituting in each possible value $x = 0, 1, 2, \ldots, 36$ and checking if $x^3 - 5x + 8$ is a square modulo 37, we find that $E(\mathbb{F}_{37})$ consists of the following 45 points modulo 37:

$$\begin{gathered}(1,\pm 2),(5,\pm 21),(6,\pm 3),(8,\pm 6),(9,\pm 27),(10,\pm 25),\\ (11,\pm 27),(12,\pm 23),(16,\pm 19),(17,\pm 27),(19,\pm 1),(20,\pm 8),\\ (21,\pm 5),(22,\pm 1),(26,\pm 8),(28,\pm 8),(30,\pm 25),(31,\pm 9),\\ (33,\pm 1),(34,\pm 25),(35,\pm 26),(36,\pm 7),\mathcal{O}.\end{gathered}$$

There are nine points of order dividing three, so as an abstract group,

$$E(\mathbb{F}_{37}) \cong C_3 \times C_{15}.$$

Theorem. Working over a finite field, the group of points $E(\mathbb{F}_p)$ is always either a cyclic group or the product of two cyclic groups.

Computing Large Multiples of a Point

To use the finite group $E(\mathbb{F}_p)$ for Diffie-Hellman, say, we need p to be quite large ($p > 2^{160}$) and we need to compute multiples

$$mP = \underbrace{P + P + \cdots + P}_{m \text{ times}} \in E(\mathbb{F}_p)$$

for very large values of m.

We can compute mP in $O(\log m)$ steps by the usual **Double-and-Add Method**. First write

$$m = m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 + \cdots + m_r \cdot 2^r \quad \text{with } m_0, \ldots, m_r \in \{0, 1\}.$$

Then mP can be computed as

$$mP = m_0 P + m_1 \cdot 2P + m_2 \cdot 2^2 P + \cdots + m_r \cdot 2^r P,$$

where $2^k P = 2 \cdot 2 \cdots 2P$ requires only k doublings.

The Algebra of Elliptic Curves

Computing Large Multiples of a Point (continued)

Thus on average, it takes approximately $\log_2(m)$ doublings and $\frac{1}{2}\log_2(m)$ additions to compute mP.

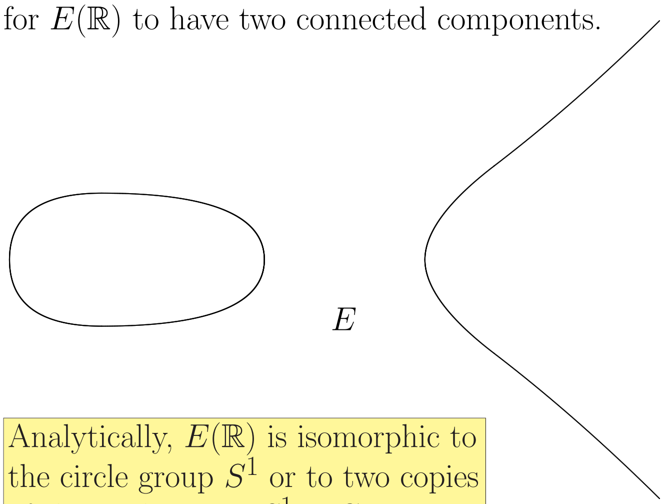
There is a simple way to reduce the computation time even further. Since it takes the same amount of time to subtract two point as it does to add two points, we can instead look at a "ternary expansion of m, which means writing

$$m = m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 + \cdots + m_r \cdot 2^r \quad \text{with } m_0, \ldots, m_r \in \{-1, 0, 1\}.$$

On average, this can be done with approximately $\frac{2}{3}$ of the m_i's equal to 0, which reduces the average number of additions to $\frac{1}{3}\log_2(m)$.

What Does $E(\mathbb{R})$ Look Like?

We have seen a picture of an $E(\mathbb{R})$. It is also possible for $E(\mathbb{R})$ to have two connected components.



Analytically, $E(\mathbb{R})$ is isomorphic to the circle group S^1 or to two copies of the circle group $S^1 \times C_2$.

What Does $E(\mathbb{F}_p)$ Look Like?

The group $E(\mathbb{F}_p)$ is obviously a finite group. Indeed, it clearly has no more than $2p + 1$ points.

For each $x \in \mathbb{F}_p$, there is a "50% chance" that the value of $f(x) = x^3 + Ax + B$ is a square in $\mathbb{F}_p^*$. And if $f(x) = y^2$ is a square, then we (usually) get two points $(x, \pm y)$ in $E(\mathbb{F}_p)$. Plus there's the point $\mathcal{O}$.

Thus we might expect $E(\mathbb{F}_p)$ to contain approximately

$$\#E(\mathbb{F}_p) \approx \tfrac{1}{2} \cdot 2 \cdot p + 1 = p + 1 \text{ points}$$

A famous theorem of Hasse makes this precise:

Theorem. (Hasse, 1922) Let E be an elliptic curve

$$y^2 = x^3 + Ax + B \quad \text{with } A, B \in \mathbb{F}_p.$$

Then
$$\left|\#E(\mathbb{F}_p) - (p + 1)\right| \le 2\sqrt{p}.$$

The Order of the Group $E(\mathbb{F}_p)$

The **Frobenius Map** is the function

$$\tau_p : E(\bar{\mathbb{F}}_p) \longrightarrow E(\bar{\mathbb{F}}_p), \qquad \tau_p(x, y) = (x^p, y^p).$$

One can check that τ_p is a **group homomorphism**.

The quantity $$a_p = p + 1 - \#E(\mathbb{F}_p)$$

is called the **Trace of Frobinius**, because one way to calculate it is to use the Frobenius map to get a linear transformation on a certain vector space $V_\ell(E)$. Then a_p is the trace of that linear transformation.

Hasse's Theorem says that

$$|a_p| \le 2\sqrt{p}.$$

For cryptography, we need $E(\mathbb{F}_p)$ to contain a subgroup of large **prime** order. How does $\#E(\mathbb{F}_p)$ vary for different E?

The Distribution of the Trace of Frobenius

There are approximately $2p$ different elliptic curves defined over $\mathbb{F}_p$.

If the $a_p(E)$ values for different E were uniformly distributed in the interval from $-2\sqrt{p}$ to $2\sqrt{p}$ then we would expect each value to appear approximately $\frac{1}{2}\sqrt{p}$ times.

This is not quite true, but it is true that the values a_p between (say) $-\sqrt{p}$ and $\sqrt{p}$ appear quite frequently. The precise statement says that the a_p values follow a Sato-Tate distribution:

Theorem. (Birch)

$$\#\{E/\mathbb{F}_p : \alpha \le a_p(E) \le \beta\} \approx \frac{1}{\pi}\int_\alpha^\beta \sqrt{4p - t^2}\, dt.$$

Computing the Order of $E(\mathbb{F}_p)$

If p is small, we can compute $x^3 + Ax + B$ for each $p = 0, 1, \dots, p-1$ and use quadratic reciprocity to check if it is a square modulo p. This takes time $O(p \log p)$.

Schoof found a deterministic polynomial-time algorithm that computes $E(\mathbb{F}_p)$ in time $O(\log p)^6$.

Elkies and Atkin made Schoof's algorithm more efficient (but probabilistic), so it is now called the

SEA Algorithm.

The details of SEA are somewhat complicated. Roughly, one studies the set of all maps of a fixed degree ℓ from E to other elliptic curves. These correspond to quotient curves E/Φ for finite subgroups $\Phi \subset E$ of order ℓ. One deduces information about a_p modulo ℓ, from which a_p can be reconstructed.

Elliptic Curve Discrete Logarithm Problem ECDLP

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$.

$$E : y^2 = x^3 + Ax + B \qquad A, B \in \mathbb{F}_p.$$

Let S and T be points in $E(\mathbb{F}_p)$. Find an integer m so that

$$T = mS.$$

Recall that the (smallest) integer m with this property is called the **Discrete Logarithm** (or **Index**) of T with respect to S and is denoted:

$$m = \log_S(T) = \mathrm{ind}_S(T).$$

Let n be the order of S in the group $E(\mathbb{F}_p)$. Then

$$\log_S : (\text{Subgroup of } E \text{ generated by } S) \longrightarrow \mathbb{Z}/n\mathbb{Z}.$$

is a group isomorphism, the inverse of $m \mapsto mS$.

How To Solve the ECDLP

Exhaustive Search Method

Compute $m_1S, m_2S, m_3S, \ldots$ for randomly chosen values m_1, m_2, m_3 until you find a multiple with $mS = T$. Expected running time is $O(p)$, since $\#E(\mathbb{F}_p) = O(p)$.

Collision Search Method

Compute two lists for randomly chosen values $m_1, m_2, \ldots$

List 1: $m_1S,\ m_2S,\ m_3S, \ldots$
List 2: $T - m_1S,\ T - m_2S,\ T - m_3S \ldots$

until finding a collision

$$m_iS = T - m_jS.$$

Expected running time is $O(\sqrt{p}\,)$ by the birthday paradox.

How To Solve the ECDLP

Pollard's ρ Method

- The collision method has running time $O(\sqrt{p}\,)$, but it takes about $O(\sqrt{p}\,)$ space to store the two lists.
- Pollards ρ method for discrete logs achieves the same $O(\sqrt{p}\,)$ running time while only requiring a very small amount of storage.
- The idea is to traverse a "random" path through the multiples $mS + nT$ until finding a collision. This path will consist of a loop with a tail attached (just like the letter ρ!!).
- It takes $O(\sqrt{p}\,)$ steps to arrive on the loop part. Then we can detect a collision in $O(\sqrt{p}\,)$ steps by storing only a small proportion of the visited points. We choose which points to store using a criterion that is independent of the underlying group law.

How Else Can DLP Be Solved?

Pollard's ρ method works for most discrete log problems.

For an abstract finite group G whose group law is given by a black box, one can **prove** that the fastest solution to the DLP has running time $O(\sqrt{\#G}\,)$.

But for specific groups with known structure, there are often faster algorithms.

- For $\mathbb{Z}/N\mathbb{Z}$, the DLP is inversion modulo N. It takes $O(\log N)$ steps by the Euclidean algorithm.
- For $\mathbb{R}^*$, the DLP can be solved using the standard logarithm,

$$\text{if } \beta = \alpha^m, \text{ then } m = \log(\beta)/\log(\alpha).$$

- For $\mathbb{F}_p^*$, there is a subexponential algorithm called the **Index Calculus** that runs in (roughly)

$$O\big(e^{c\sqrt[3]{\log p}}\big) \text{ steps.}$$

Does ECDLP Have a Faster Solution?

The principal reason that elliptic curve groups are used for cryptography is:

For general elliptic curves, the fastest known method to solve ECDLP is Pollard's ρ Method!!

This means that it is not currently feasible to solve ECDLP in $E(\mathbb{F}_q)$ if (say) $q > 2^{160}$.

A DLP of equivalent difficulty in $\mathbb{F}_q^*$ requires $q \approx 2^{1000}$. Similarly, ECDLP with $q \approx 2^{160}$ is approximately as hard as factoring a 1000 bit number.

Hence cryptographic constructions based on ECDLP have smaller keys, smaller message blocks, and may also be faster.

Solving ECDLP in Special Cases

For "most" elliptic curves, the best known solution to ECDLP has running time $O(\sqrt{p})$. But for certain special classes of curves, there are faster methods.

It is important to know which curves have fast ECDLP algorithms so that we can avoid using them.

Elliptic Curves $E(\mathbb{F}_p)$ With Exactly p Points

If $\#E(\mathbb{F}_p) = p$, then there is a "p-adic logarithm map" that gives an easily computed homomorphism

$$\log_{p\text{-adic}} : E(\mathbb{F}_p) \longrightarrow \mathbb{Z}/p\mathbb{Z}.$$

It is easy to solve the discrete logarithm problem in $\mathbb{Z}/p\mathbb{Z}$, so if $\#E(\mathbb{F}_p) = p$, then we can solve ECDLP in time $O(\log p)$.

5.1. **The Edwards form for elliptic curves.** In 2007 the American mathematician Harold M. Edwards proposed to use a different equation instead of the equation for elliptic curves as given in the previous section. An advantage of this new form is, that under mild additional conditions the group law on the set of solutions to this equation, is given by a single formula rather than a case-by-case consideration. Immediately after Edwards's proposal others picked it up, for example Dan Bernstein (Chicago) and Tanja Lange (Eindhoven). Chapters 3–5 of the bachelor's thesis of Marion Dam (2012): http://irs.ub.rug.nl/dbi/503b69f805f96 provide a detailed discussion of the idea by Edwards, and its relation to elliptic curves. We briefly describe it here and refer to Dam's text for proofs.

Suppose K is a field. The *circle group over* K, denoted $C(K)$, is defined as

$$C(K) := \left\{(x, y) \in K \times K \; ; \; x^2 + y^2 = 1\right\}.$$

This set is made into a group as follows: let $O := (1, 0) \in C(K)$. Given two points $P_j = (x_j, y_j) \in C(K)$, define

$$P_1 + P_2 := (x_1x_2 - y_1y_2, x_1y_2 + x_2y_1).$$

It is not difficult to verify that this provides $C(K)$ with the structure of an abelian group, with unit element O and inversion $-(x, y) = (x, -y)$.

In the special case $K = \mathbb{R}$ the points (x, y) in $C(\mathbb{R})$ correspond to the points $x + yi$ on the unit circle in $\mathbb{C}$. The usual multiplication in this unit circle in this way yields the addition on $C(\mathbb{R})$ as defined here. This is the reason why in general $C(K)$ is called the circle group (over K).

If $1 + 1 \neq 0$ in K, i.e., if $-1 \neq 1$ in K, then $P := (0, 1) \in C(K)$ satisfies $P + P = (-1, 0)$ and $P + P + P = (0, -1)$ and $4P = O$. So in this case $C(K)$ contains a point of order 4, namely P.

The Edwards form can be regarded as a variation on the circle group. Namely, let $d \in K$. Define $C_d(K)$ as

$$C_d(K) := \left\{(x, y) \in K \times K\ ;\ x^2 + y^2 = 1 + dx^2y^2\right\}.$$

For $d = 0$ this is the circle group. And for every d, the set $\{(\pm 1, 0), (0, \pm 1)\}$ is in $C_d(K)$. We now try to make $C_d(K)$ into a group. Let $P_j = (x_j, y_j) \in C_d(K)$. Put

$$P_1 + P_2 := \left(\frac{x_1x_2 - y_1y_2}{1 + dx_1x_2y_1y_2}, \frac{x_1y_2 + x_2y_1}{1 - dx_1x_2y_1y_2}\right),$$

provided this is defined, i.e., the denominators are non-zero. A long but straightforward computation shows that *if* $P_1 + P_2$ is defined, then it is an element of $C_d(K)$. Note that the special case $d = 0$ agrees with the addition formula on the circle. In case $d \neq 0$ is not a square in K, it can be shown that the denominators appearing above are non-zero for all $P_1, P_2 \in C_d(K)$. So in this case the addition is defined on all of $C_d(K)$. It turns out that this makes $C_d(K)$ into an abelian group. The inverse of a point $P = (x, y) \in C_d(K)$ is $-P := (x, -y)$. Exactly as in the case of the circle group, $P := (0, 1) \in C_d(K)$ satisfies $P + P = (-1, 0)$ and $P + P + P = (0, -1)$ and $4P = O$. So (recall we assume $-1 \neq 1$ in K) the point P has order 4 in $C_d(K)$.

The bachelor's thesis of Marion Dam explains the relation between this Edwards form and elliptic curves. This starts from the observation that one should look for elliptic curves containing a point (over K) of order 4. Given such an elliptic curve E, Dam describes an explicit bijection between $E(K)$ and $C_d(K)$ for some $d \in K$. This transports the group structure on E to that on C_d. Vice versa, given $d \neq 0, \neq 1$, she describes an elliptic curve E such that $E(K)$ contains a point of order 4, and a bijection $C_d(K) \to E(K)$ which is the inverse of the one above. This explains how the group law formula on C_d can be constructed, and why in fact it is a group law. Moreover, it shows what kind of groups can be obtained as $C_d(K)$: precisely all elliptic curve groups containing a point of order 4.

We finish these lecture notes with some lines of Magma code, illustrating the connection between the Edwards form and the standard way of representing elliptic curves.

```
q:=RandomPrime(15);
Fq:=GF(q);
P2<x,y,z>:=ProjectiveSpace(Fq,2);
d:=Random(Fq);
Cd:=Curve(P2, z^2*(x^2+y^2)-z^4-d*x^2*y^2);
Pt:=Cd![1,0,1];
set:=Points(Cd);
a:=Cd![1,0,0]; b:=Cd![0,1,0];
E,f:=EllipticCurve(Cd,Pt);
P1:=Random(set); P2:=Random(set);
foo,fi:=IsInvertible(f);
fi(f(P1)+f(P2));
```

The following code first defines the Edwards curve C_d for a variable d, and constructs an elliptic curve E, a 'map' $f : C_d \to E$ and an 'inverse' $fi : E \to C_d$. Then two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on C_d are constructed, in which x_1, x_2 are two variables, and the y_j are taken in such a way that indeed $P_1, P_2 \in C_d$. Finally, P_1 and P_2 are added by first sending them to E by means of the map f, then the images are added using the group structure of E, and finally the sum in E is sent back to C_d by means of the map fi.

The calculation, which takes quite some time since apparently Magma has difficulty calculating the inverse of f, reveals that the formula for adding points on C_d equals the one obtained as described here.

```
Qd<d>:=FunctionField(Rationals());
P2<x,y,z>:=ProjectiveSpace(Qd,2);
Cd:=Curve(P2, z^2*(x^2+y^2)-z^4-d*x^2*y^2);

Qdx1<x1>:=FunctionField(Qd);
R<Y>:=PolynomialRing(Qdx1);
K<y1>:=ext<Qdx1 | x1^2+Y^2-1-d*x1^2*Y^2>;
K2<x2>:=FunctionField(K);
S<T>:=PolynomialRing(K2);
L<y2>:=ext<K2 | x2^2+T^2-1-d*x2^2*T^2>;

Cd:=BaseChange(Cd, L);
Pt:=Cd![1,0,1];
E,f:=EllipticCurve(Cd,Pt);
P1:=Cd![x1,y1,1]; P2:=Cd![x2,y2,1];
foo,fi:=IsInvertible(f);
fi(f(P1)+f(P2));
```

6. Exercises on security

(1) In the AES, the S-Box uses a field consisting of 256 elements. In the same spirit, construct fields consisting of 4 and also of 8 elements.

(2) Given the map λ used in the AES, find a formula for $\lambda^2 = \lambda \circ \lambda$ and show that λ^4 equals the identity map.

(3) Given $m = x^8 + x^4 + x^3 + x + 1$, compute the inverse of $x^2 + 1 \bmod m$ in $\mathbb{F}_2[x]/(m)$.

(4) Compute the S-box image $\sigma(f)$ and the pre-image $\sigma^{-1}(f)$, for $f = x^2 + 1 \bmod m$.

(5) The integer $n = 561$ has the property that it is composite. Yet, just as in Fermat's little theorem for prime numbers, it satisfies $a^{n-1} \equiv 1 \bmod n$ whenever a and n are coprime. Prove that $n = 561$ indeed has the asserted properties.

(6) For each of the primes $p < 20$, find a primitive root modulo p.

(7) If p is prime and g is a primitive root modulo p, what is the discrete logarithm of $-1 \bmod p$ with respect to g?

(8) In this exercise we use that 2 is a primitive root modulo 29.
   (a) Note that $2^5 \equiv 3 \bmod 29$. What is the discrete log of $3 \bmod 29$ w.r.t. 2?
   (b) Now observe $2 \cdot 3 \cdot 5 \equiv 1 \bmod 29$, and deduce from this the discrete log of $5 \bmod 29$ w.r.t. 2.
   (c) Use $7 \cdot 4$ to find the discrete log of $7 \bmod 29$ w.r.t. 2.
   (d) Find the discrete log of $11 \bmod 29$ w.r.t. 2, and also of $13 \bmod 29 = -16 \bmod 29$.

(9) Using the previous exercise, find the prime numbers $p < 29$ such that $p \bmod 29$ is a square modulo 29.

(10) Given that 2 is a square modulo 31 and 3 is not, use the Tonelli-Shanks algorithm to find a square root of 2 modulo 31.

(11) In the special cases $n = 9$, $n = 15$, and $n = 21$, compute the cardinality of the set A mentioned in the discussion about Miller-Rabin.

What is this set A in the case that $n = p$ is an odd prime number?

(12) Lemma 3.2 shows that for p an odd prime and $e > 0$, the group $(\mathbb{Z}/p^e\mathbb{Z})^*$ is *cyclic*, i.e., it consists of the powers of one generator $g \bmod p^e$.
  (a) Show that if a group is cyclic, then it contains at most 1 element of order 2.
  (b) Show that if $e \geq 3$, then $(\mathbb{Z}/2^e\mathbb{Z})^*$ contains more than one element of order 2, hence this group is *not* cyclic. And what happens for $e = 2$ and for $e = 1$?

(13) Here are two 'baby'-examples of a Pollard $p - 1$ factorisation.
  (a) Take $n = 1001$ and $E = 6$. What is $\gcd(n, 2^E - 1)$?
  (b) Now take $n = 10001$ and $E = 12$. Show that $\gcd(n, 2^E - 1) = 1$, and that $\gcd(n, 3^E - 1)$ produces a nontrivial factor of n.

(14) Let p be a prime satisfying $p \equiv 3 \bmod 4$. Take any $a \in \mathbb{F}_p$ with $a \neq 0$. Consider E over $\mathbb{F}_p$ corresponding to the equation $y^2 = x^3 + ax$.
  (a) Verify that E defines an elliptic curve over $\mathbb{F}_p$.
  (b) Show that $(-1)^{(p-1)/2} = -1$ and conclude that -1 is not a square modulo p.
  (c) Write $f(x) := x^3 + ax$. For $b \in \mathbb{F}_p$, show that either $f(b) = 0$, or exactly one of $f(b), f(-b)$ is a nonzero square in $\mathbb{F}_p$.
  (d) Conclude that $E(\mathbb{F}_p)$ is a group consisting of precisely $p+1$ elements.
  (e) Now take $p = 67$. How can you make sure that the group considered here, is generated by one element?

(15) Given a point $Q = (a, b)$ with $a = 0$ or $b = 0$ in an Edwards group $C_d(K)$, what is the formula describing 'translation by Q' in $C_d(K)$?

(16) Suppose K is a field in which $1 + 1 \neq 0$, and $d \neq 0$ is an element of K which is not a square. As claimed in the lecture notes, these conditions ensure that the set $C_d(K)$ with the given addition formula is a group. Find *all* elements of order 2 in this group, and also *all* elements of order 4.