Network Security

Chapter 4
Asymmetric Cryptography

"However, prior exposure to discrete mathematics will help the reader to appreciate the concepts presented here."

E. Amoroso in another context [Amo94] *:o)*

Asymmetric Cryptography (1)

- General idea:
  - Use two different keys *-K* and *+K* for encryption and decryption
  - Given a random ciphertext $c = E(+K, m)$ and *+K* it should be infeasible to compute $m = D(-K, c) = D(-K, E(+K, m))$
    - This implies that it should be infeasible to compute *-K* when given *+K*
  - The key *-K* is only known to one entity A and is called A's *private key* $-K_A$
  - The key *+K* can be publicly announced and is called A's *public key* $+K_A$
- Applications:
  - Encryption:
    - If B encrypts a message with A's public key $+K_A$, he can be sure that only A can decrypt it using $-K_A$
  - Signing:
    - If A encrypts a message with his own private key $-K_A$, everyone can verify this signature by decrypting it with A's public key $+K_A$
  - Attention: It is crucial, that everyone can verify that he really knows A's public key and not the key of an adversary!

Asymmetric Cryptography (2)

- Design of asymmetric cryptosystems:
  - Difficulty: Find an algorithm and a method to construct two keys *-K, +K* such that it is not possible to decipher *E(+K, m)* with the knowledge of *+K*
  - Constraints:
    - The key length should be "manageable"
    - Encrypted messages should not be arbitrarily longer than unencrypted messages (we would tolerate a small constant factor)
    - Encryption and decryption should not consume too much resources (time, memory)
  - Basic idea: Take a problem in the area of mathematics / computer science, that is *hard* to solve when knowing only *+K*, but *easy* to solve when knowing *-K*
    - Knapsack problems: basis of first working algorithms, which were unfortunately almost all proven to be insecure
    - Factorization problem: basis of the RSA algorithm
    - Discrete logarithm problem: basis of Diffie-Hellman and ElGamal

Some Mathematical Background (1)

- Definitions:
  - Let $\mathbb{Z}$ be the number of integers, and $a, b, n \in \mathbb{Z}$
  - We say *a divides b* ("$a \mid b$") if there exists an integer $k \in \mathbb{Z}$ such that $a \times k = b$
  - We say *a is prime* if it is positive and the only divisors of *a* are 1 and *a*
  - We say *r* is the *remainder* of *a* divided by *n* if $r = a - \lfloor a / n \rfloor \times n$ where $\lfloor x \rfloor$ denotes the largest integer less than or equal to *x*
    - Example: 4 is the remainder of 11 divided by 7 as $4 = 11 - \lfloor 11 / 7 \rfloor \times 7$
    - We can write this in another way: $a = q \times n + r$ with $q = \lfloor a / n \rfloor$
  - For the remainder *r* of the division of *a* by *n* we write *a MOD n*
  - We say *b is congruent a mod n* if it has the same remainder like *a* when divided by *n*. So, *n* divides *(a-b)*, and we write $b \equiv a \bmod n$
    - Examples: $4 \equiv 11 \bmod 7$, $25 \equiv 11 \bmod 7$, $11 \equiv 25 \bmod 7$, $11 \equiv 4 \bmod 7$, $-10 \equiv 4 \bmod 7$
  - As the remainder *r* of division by *n* is always smaller than *n*, we sometimes represent the set $\{x \text{ MOD } n \mid x \in \mathbb{Z}\}$ by elements of the set $\mathbb{Z}_n = \{0, 1, ..., n - 1\}$

Some Mathematical Background (2)

Properties of Modular Arithmetic	
Property	Expression
Commutative Laws	$(a + b) \text{ MOD } n = (b + a) \text{ MOD } n$
	$(a \times b) \text{ MOD } n = (b \times a) \text{ MOD } n$
Associative Laws	$[(a + b) + c] \text{ MOD } n = [a + (b + c)] \text{ MOD } n$
	$[(a \times b) \times c] \text{ MOD } n = [a \times (b \times c)] \text{ MOD } n$
Distributive Law	$[a \times (b + c)] \text{ MOD } n = [(a \times b) + (a \times c)] \text{ MOD } n$
Identities	$(0 + a) \text{ MOD } n = a \text{ MOD } n$
	$(1 \times a) \text{ MOD } n = a \text{ MOD } n$
Inverses	$\forall a \in \mathbb{Z}_n: \exists (-a) \in \mathbb{Z}_n : a + (-a) \equiv 0 \bmod n$
	p is prime $\Rightarrow \forall a \in \mathbb{Z}_p: \exists (a^{-1}) \in \mathbb{Z}_p: a \times (a^{-1}) \equiv 1 \bmod p$

Some Mathematical Background (3)

- Greatest common divisor:
  - $c = gcd(a, b) :\Leftrightarrow (c \mid a) \wedge (c \mid b) \wedge [\forall d: (d \mid a) \wedge (d \mid b) \Rightarrow (d \mid c)]$ and $gcd(a, 0) := |a|$
- The *gcd recursion theorem*:
  - $\forall a, b \in \mathbb{Z}^+: gcd(a, b) = gcd(b, a \text{ MOD } b)$
  - Proof:
    - As *gcd(a, b)* divides both *a* and *b* it also divides any linear combination of them, especially $(a - \lfloor a / b \rfloor \times b) = a \text{ MOD } b$, so $gcd(a, b) \mid gcd(b, a \text{ MOD } b)$
    - As $gcd(b, a \text{ MOD } b)$ divides both *b* and *a* MOD *b* it also divides any linear combination of them, especially $\lfloor a / b \rfloor \times b + (a \text{ MOD } b) = a$, so $gcd(b, a \text{ MOD } b) \mid gcd(a, b)$
- Euclidean Algorithm:
  - The algorithm *Euclid* given *a, b* computes gcd(a, b)
  - ```
    int Euclid(int a, b)
    {   if   (b = 0)   { return(a);}
                       { return(Euclid(b, a MOD b);} }
    ```
    ```

Some Mathematical Background (4)

- Extended Euclidean Algorithm:
  - The algorithm *ExtendedEuclid* given *a, b* computes *d, m, n* such that: $d = gcd(a, b) = m \times a + n \times b$
  - *struct{int d, m, n}   ExtendedEuclid(int a, b)*
    { int d, d', m, m', n, n';
    if (b = 0) {return(a, 1, 0); }
    (d', m', n') = ExtendedEuclid(b, a MOD *b);*
    $(d, m, n) = (d', n', m' - \lfloor a / b \rfloor \times n');$
    return(d, m, n); }
  - Proof: (by induction)
    - Basic case (*a*, 0): $gcd(a, 0) = a = 1 \times a + 0 \times 0$
    - Induction from (b, a MOD b) to (a, b):
      - ExtendedEuclid computes d', m', n' correctly (induction hypothesis)
      - $d = d' \quad = m' \times b + n' \times (a \text{ MOD } b) = m' \times b + n' \times (a - \lfloor a / b \rfloor \times b)$
        $= n' \times a + (m' - \lfloor a / b \rfloor \times n') \times b$
  - The run time of Euclid(a, b) and ExtendedEuclid(a, b) is of O(log b)
    - Proof: see [Cor90a], section 33.2

Some Mathematical Background (5)

- Summarizing the discussion of the Euclidean algorithms we have:

  Lemma 1:

  Let $a, b \in \mathbb{N}$ and d = gcd(a, b). Then there exists $m, n \in \mathbb{N}$ such that:
  $d = m \times a + n \times b$
- We can use this lemma to prove the following:

  Theorem 1 (Euclid):

  If a prime divides the product of two integers, then it divides at least one of the integers: $p \mid (a \times b) \Rightarrow (p \mid a) \vee (p \mid b)$
  - Proof: Let $p \mid (a \times b)$
    - If $p \mid a$ then we are done.
    - If not then $gcd(p, a) = 1 \Rightarrow$
      $\exists\, m, n \in \mathbb{N}: \quad 1 = m \times p + n \times a$
      $\Leftrightarrow \quad b = m \times p \times b + n \times a \times b$
      As $p \mid (a \times b)$, *p* divides both summands of the equation and so it divides also the sum which is *b* ■

Some Mathematical Background (6)

- A small, but nice excursion:
  - With the help of Theorem 1 the proof that $\sqrt{2}$ is not a rational number can be given in a very elegant way:

    Assume that $\sqrt{2}$ can be expressed as a rational number *m* / *n* and that this fraction has been reduced such that *gcd(m, n)* = 1:

    $$\Rightarrow \quad \sqrt{2} = \frac{m}{n} \quad \Leftrightarrow \quad 2 = \frac{m^2}{n^2} \quad \Leftrightarrow \quad 2n^2 = m^2$$

    So, 2 divides m^2, and thus by Theorem 1 it also divides *m*, and so 4 divides m^2. But then 4 divides $2n^2$ and, therefore, 2 divides also n^2.

    Again by Theorem 1 this implies that 2 divides *n* and so 2 divides both *m* and *n*, which is a contradiction to the assumption that the fraction *m* / *n* is reduced.

    ■

  - And now to something more useful... – for cryptography :o)

Some Mathematical Background (7)

Theorem 2 (fundamental theorem of arithmetic):

Factorization into primes is unique up to order.

- Proof:
  - We will show that every integer with a non-unique factorization has a proper divisor with a non-unique factorization which leads to a clear contradiction when we finally have reduced to a prime number.
  - Let's assume that n is an integer with a non-unique factorization:

    $n = p_1 \times p_2 \times ... \times p_r$

    $ = q_1 \times q_2 \times ... \times q_s$

    The primes are not necessarily distinct, but the second factorization is not simply a reordering of the first one.

    As p_1 divides *n* it also divides the product $q_1 \times q_2 \times ... \times q_s$. By repeated application of Theorem 1 we show that there is at least one q_i which is divisible by p_1. If necessary reorder the q_i's so that it is q_1. As both p_1 and q_1 are prime they have to be equal. So we can divide by p_1 and we have that n / p_1 has a non-unique factorization.

Some Mathematical Background (8)

- We will use Theorem 2 to prove the following

  Corollary 1:

  If $gcd(c, m) = 1$ and $(a \times c) \equiv (b \times c) \bmod m$, then $a \equiv b \bmod m$

  - Proof: As $(a \times c) \equiv (b \times c) \bmod m \Rightarrow \exists\, n \in \mathbb{N}: (a \times c) - (b \times c) = n \times m$

    $$\Leftrightarrow \underbrace{(a - b)}_{p_1 \times \ldots \times p_i} \times \underbrace{c}_{q_1 \times \ldots \times q_j} = \underbrace{n}_{r_1 \times \ldots \times r_k} \times \underbrace{m}_{s_1 \times \ldots \times s_l}$$

    $$\Leftrightarrow p_1 \times \ldots \times p_i \times q_1 \times \ldots \times q_j = r_1 \times \ldots \times r_k \times s_1 \times \ldots \times s_l$$

    Please note that the p's, q's, r's and s's are prime and do not need to be distinct, but as gcd(c, m) = 1, there are no indices g, h such that $q_g = s_h$.

    So we can continuously divide the equation by all q's without ever "eliminating" one s and will finally end up with something like

    $$\Leftrightarrow p_1 \times \ldots \times p_i = r_1 \times \ldots \times r_o \times s_1 \times \ldots \times s_l$$

    (note that there will be fewer r's)

    $$\Leftrightarrow (a - b) = r_1 \times \ldots \times r_o \times m$$

    $$\Rightarrow a \equiv b \bmod m$$

    ■

Some Mathematical Background (9)

- Let $\Phi(n)$ denote the number of positive integers less than n and relatively prime to n
  - Examples: $\Phi(4) = 2$, $\Phi(6) = 2$, $\Phi(7) = 6$, $\Phi(15) = 8$
  - If p is prime $\Rightarrow \Phi(p) = p - 1$

  Theorem 3 (Euler):

  Let n and b be positive and relatively prime integers, i.e. $gcd(n, b) = 1$
  $\Rightarrow b^{\Phi(n)} \equiv 1 \bmod n$

  Proof:

  - Let $t = \Phi(n)$ and $a_1, \ldots a_t$ be the positive integers less than n which are relatively prime to n.
    Define $r_1, \ldots, r_t$ to be the residues of $b \times a_1 \bmod n, \ldots, b \times a_t \bmod n$ that is to say: $b \times a_i \equiv r_i \bmod n$.
  - Note that $i \neq j \Rightarrow r_i \neq r_j$.
    If this would not hold, we would have $b \times a_i \equiv b \times a_j \bmod n$
    and as gcd(b, n) = 1, Corollary 1 would imply $a_i \equiv a_j$ mod n which can not be as a_i and a_j are by definition distinct integers between 0 and n

Some Mathematical Background (10)

Proof (continued):

- We also know that each r_i is relatively prime to n because any common divisor k of r_i and n, i.e. $n = k \times m$ and $r_i = p_i \times k$, would also have to divide a_i,
- as $b \times a_i \equiv (p_i \times k) \bmod (k \times m) \Rightarrow \exists\, s \in \mathbb{N}$: $(b \times a_i) - (p_i \times k) = s \times k \times m$

  $\Leftrightarrow (b \times a_i) = s \times k \times m + (p_i \times k)$

  Because k divides each of the summands on the right-hand side and k does not divide b by assumption (n and b are relatively prime), it would also have to divide a_i which is supposed to be relatively prime to n
- Thus $r_1, ..., r_t$ is a set of $\Phi(n)$ distinct integers which are relatively prime to n. This means that they are exactly the same as $a_1, ... a_t$, except that they are in a different order. In particular, we know that $r_1 \times ... \times r_t = a_1 \times ... \times a_t$
- We now use the congruence

  $$r_1 \times ... \times r_t \equiv b \times a_1 \times ... \times b \times a_t \bmod n$$
  $$\Leftrightarrow r_1 \times ... \times r_t \equiv b^t \times a_1 \times ... \times a_t \bmod n$$
  $$\Leftrightarrow r_1 \times ... \times r_t \equiv b^t \times r_1 \times ... \times r_t \bmod n$$
- As all r_i are relatively prime to n we can use Corollary 1 and divide by their product giving: $1 \equiv b^t \bmod n \Leftrightarrow 1 \equiv b^{\Phi(n)} \bmod n$ ■

Some Mathematical Background (11)

Theorem 4 (Chinese Remainder Theorem):

Let $m_1, ..., m_r$ be positive integers that are pairwise relatively prime, i.e. $\forall\, i \neq j$: $gcd(m_i, m_j) = 1$. Let $a_1, ..., a_r$ be arbitrary integers. Then there exists an integer a such that:

$$a \equiv a_1 \bmod m_1$$
$$a \equiv a_2 \bmod m_2$$
$$...$$
$$a \equiv a_r \bmod m_r$$

Furthermore, a is unique modulo $M := m_1 \times ... \times m_r$

Proof:

- For all $i \in \{1, .., r\}$ we define $M_i := (M / m_i)^{\Phi(m_i)}$
- As M_i is by definition relatively prime to m_i we can apply Theorem 3 and know that $M_i \equiv 1 \bmod m_i$
- Since M_i is divisible by m_j for every $j \neq i$, we have $\forall\, j \neq i : M_i \equiv 0 \bmod m_j$

Some Mathematical Background (12)

Proof (continued):

- We can now construct the solution by defining:

  $a := a_1 \times M_1 + a_2 \times M_2 + ... + a_r \times M_r$

- The two arguments given above concerning the congruences of the M_i imply that *a* actually satisfies all of the congruences.
- To see that *a* is unique modulo *M*, let *b* be any other integer satisfying the *r* congruences. As $a \equiv c \bmod n$ and $b \equiv c \bmod n \Rightarrow a \equiv b \bmod n$

  we have $\forall i \in \{1, .., r\}: a \equiv b \bmod m_i$

  $\Rightarrow \forall i \in \{1, .., r\}: m_i \mid (a - b)$

  $\Rightarrow M \mid (a-b)$ as the m_i are pairwise relatively prime

  $\Leftrightarrow a \equiv b \bmod M$ ∎

Some Mathematical Background (13)

<u>Lemma 2:</u>

If gcd(m, n) = 1, then $\Phi(m \times n) = \Phi(m) \times \Phi(n)$

Proof:

- Let *a* be a positive integer less than and relatively prime to $m \times n$. In other words, *a* is one of the integers counted by $\Phi(m \times n)$.
- Consider the correspondence $a \rightarrow (a \text{ MOD } m, a \text{ MOD } n)$

  The integer *a* is relatively prime to *m* and relatively prime to *n* (if not it would divide $m \times n$).

  So, (*a* MOD *m*) is relatively prime to *m* and (*a* MOD *n*) is relatively prime to *n* as: $a = \lfloor a / m \rfloor \times m + (a \text{ MOD } m)$, so if there would be a common divisor of *m* and (*a* MOD *m*), this divisor would also divide *a.*

  Thus every number *a* counted by $\Phi(m \times n)$ corresponds to a pair of two integers (*a* MOD *m, a* MOD *n*), the first one counted by $\Phi(m)$ and the second one counted by $\Phi(n)$.

Some Mathematical Background (14)

Proof (continued):

- Because of the second part of Theorem 4, the uniqueness of the solution *a* modulo (m × n) to the simultaneous congruences:

  $a \equiv (a\ MOD\ m)\ mod\ m$
  $a \equiv (a\ MOD\ n)\ mod\ n$

  we can deduce, that distinct integers counted by $\Phi(m \times n)$ correspond to distinct pairs:

  - Too see this, suppose that a ≠ b counted by $\Phi(m \times n)$ does correspond to the same pair (*a MOD m, a MOD n).* This leads to a contradiction as b would also fulfill the congruences:

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

    but the solution to these congruences is unique modulo (m × n)

  Therefore, $\Phi(m \times n)$ is at most the number of such pairs:

  $\Phi(m \times n) \leq \Phi(m) \times \Phi(n)$ ■

Some Mathematical Background (15)

Proof (continued):

- Consider now a pair of integers (*b*, *c*), one counted by $\Phi(m)$ and the other one counted by $\Phi(n)$:

  Using the first part of Theorem 4 we can construct a unique positive integer *a* less than and relatively prime to $m \times n$:

  $a \equiv b\ mod\ m$
  $a \equiv c\ mod\ n$

  So, the number of such pairs is at most $\Phi(m \times n)$:

  $\Phi(m \times n) \geq \Phi(m) \times \Phi(n)$ ■

The RSA Public Key Algorithm (1)

- The RSA algorithm was invented in 1977 by R. Rivest, A. Shamir and L. Adleman [RSA78] and is based on Theorem 3.
- Let *p, q* be distinct large primes and $n = p \times q$. Assume, we have also two integers *e* and *d* such that:

  $d \times e \equiv 1 \bmod \Phi(n)$
- Let *M* be an integer that represents the message to be encrypted, with *M* positive, smaller than and relatively prime to *n*.
  - Example: Encode with <blank> = 99, A = 10, B = 11, ..., Z = 35
    So "HELLO" would be encoded as 1714212124.
    If necessary, break M into blocks of smaller messages: 17142 12124
- To encrypt, compute: $E = M^e \text{ MOD } n$
  - This can be done efficiently using the *square-and-multiply algorithm*
- To decrypt, compute: $M' = E^d \text{ MOD } n$
  - As $d \times e \equiv 1 \bmod \Phi(n) \quad \Rightarrow \exists\, k \in \mathbb{Z}: \quad (d \times e) - 1 = k \times \Phi(n)$

    $\Leftrightarrow (d \times e) = k \times \Phi(n) + 1$

    we have: $M' \equiv E^d \equiv M^{(e \times d)} \equiv M^{(k \times \Phi(n) + 1)} \equiv 1^k \times M \equiv M \bmod n$ ■

The RSA Public Key Algorithm (2)

- As $(d \times e) = (e \times d)$ the operation also works in the opposite direction, that means you can encrypt with *d* and decrypt with *e*
  - This property allows to use the same keys *d* and *e* for:
    - Receiving messages that have been encrypted with one's public key
    - Sending messages that have been signed with one's private key
- To set up a key pair for RSA:
  - Randomly choose two primes *p* and *q* (of 100 to 200 digits each)
  - Compute $n = p \times q$, $\Phi(n) = (p - 1) \times (q - 1)$ (Lemma 2)
  - Randomly choose *e*, so that $\gcd(e, \Phi(n)) = 1$
  - With the extended euclidean algorithm compute *d* and *c*, such that:

    $e \times d + \Phi(n) \times c = 1$, note that this implies, that $e \times d \equiv 1 \bmod \Phi(n)$
  - The public key is the pair *(e, n)*
  - The private key is the pair *(d, n)*

The RSA Public Key Algorithm (3)

- The security of the scheme lies in the difficulty of factoring $n = p \times q$ as it is easy to compute $\Phi(n)$ and then d, when p and q are known
- This class will not teach why it is difficult to factor large n's, as this would require to dive deep into mathematics
  - If p and q fulfill certain properties, the best known algorithms are exponential in the number of digits of n
    - Please be aware that if you choose p and q in an "unfortunate" way, there might be algorithms that can factor more efficiently and your RSA encryption is not at all secure:
      - Therefore, p and q should be about the same bitlength and sufficiently large
      - $(p - q)$ should not be too small
      - If you want to choose a small encryption exponent, e.g. 3, there might be additional constraints, e.g. $\gcd(p - 1, 3) = 1$ and $\gcd(q - 1, 3) = 1$
    - The security of RSA also depends on the primes generated being truly random (like every key creation method for any algorithm)
    - Moral: If you are to implement RSA by yourself, ask a mathematician or better a cryptographer to check your design

Diffie-Hellman Key Exchange (1)

- The Diffie-Hellman key exchange was first published in the landmark paper [DH76], which also introduced the fundamental idea of asymmetric cryptography
- The DH exchange in its basic form enables two parties A and B to agree upon a *shared secret* using a public channel:
  - *Public channel* means, that a potential attacker E (E stands for eavesdropper) can read all messages exchanged between A and B
  - It is important, that A and B can be sure, that the attacker is not able to alter messages, as in this case he might launch a *man-in-the-middle attack*
  - The mathematical basis for the DH exchange is the problem of finding *discrete logarithms in finite fields*
  - The DH exchange is *not* an asymmetric encryption algorithm, but is nevertheless introduced here as it goes well with the mathematical flavor of this lecture... :o)

Some More Mathematical Background (1)

- Definition: *finite groups*
  - A group (S, $\oplus$) is a set S together with a binary operation $\oplus$ for which the following properties hold:
    - *Closure:* For all $a, b \in S$, we have $a \oplus b \in S$
    - *Identity:* There is an element $e \in S$, such that $e \oplus a = a \oplus e = a$ for all $a \in S$
    - *Associativity:* For all $a, b, c \in S$, we have $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
    - *Inverses:* For each $a \in S$, there exists a unique element $b \in S$, such that $a \oplus b = b \oplus a = e$
  - If a group (S, $\oplus$) satisfies the commutative law $\forall\ a, b \in S$: $a \oplus b = b \oplus a$ then it is called an *Abelian group*
  - If a group (S, $\oplus$) has only a finite set of elements, i.e. $|S| < \infty$, then it is called a *finite group*

TELEMATIK Rechnernetze

Some More Mathematical Background (2)

- Examples:
  - ($\mathbb{Z}_n$, $+_n$)
    - with $\mathbb{Z}_n := \{[0]_n, [1]_n, ..., [n - 1]_n\}$
    - where $[a]_n := \{b \in \mathbb{Z} \mid b \equiv a \bmod n\}$ and
    - $+_n$ is defined such that $[a]_n +_n [b]_n = [a + b]_n$

    is a finite abelian group

    For the proof see the table showing the properties of modular arithmetic
  - ($\mathbb{Z}^*_n$, $\times_n$)
    - with $\mathbb{Z}^*_n := \{[a]_n \in \mathbb{Z}_n \mid gcd(a, n) = 1\}$, and
    - $\times_n$ is defined such that $[a]_n \times_n [b]_n = [a \times b]_n$

    is a finite Abelian group. Please note that $\mathbb{Z}^*_n$ just contains those elements of $\mathbb{Z}_n$ that have a multiplicative inverse modulo n

    For the proof see the properties of modular arithmetic
    - Example: $\mathbb{Z}^*_{15} = \{[1]_{15}, [2]_{15}, [4]_{15}, [7]_{15}, [8]_{15}, [11]_{15}, [13]_{15}, [14]_{15}\}$, as

      $1 \times 1 \equiv 1 \bmod 15$, $2 \times 8 \equiv 1 \bmod 15$, $4 \times 4 \equiv 1 \bmod 15$,

      $7 \times 13 \equiv 1 \bmod 15$, $11 \times 11 \equiv 1 \bmod 15$, $14 \times 14 \equiv 1 \bmod 15$

Some More Mathematical Background (3)

- If it is clear that we are talking about $(\mathbb{Z}_n, +_n)$ or $(\mathbb{Z}^*_n, \times_n)$ we often represent equivalence classes $[a]_n$ by their representative elements a and denote $+_n$ *and* $\times_n$ by + and ×, respectively.
- Definition: *finite fields*
  - A *field* $(S, \oplus, \otimes)$ is a set S together with two operations ⊕, ⊗ such that
    - $(S, \oplus)$ and $(S \setminus \{e_\oplus\}, \otimes)$ are commutative groups, i.e. only the identity element concerning the operation ⊕ does not need to have an inverse regarding the operation ⊗
    - For all $a, b, c \in S$, we have $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
  - If $|S| < \infty$ then $(S, \oplus, \otimes)$ is called a *finite field*
- Example:
  - $(\mathbb{Z}_p, +_p, \times_p)$ is a finite field for each prime p

Some More Mathematical Background (4)

- Definition: *primitive root, generator*
  - Let $(S, \bullet)$ be a group, $g \in S$ and $g^a := g \bullet g \bullet \ldots \bullet g$ (a times with $a \in \mathbb{Z}^+$)
    Then g is called a *primitive root* or *generator* of $(S, \bullet)$
    $:\Leftrightarrow \{g^a \mid 1 \leq a \leq |S|\} = S$
- Examples:
  - 1 is a primitive root of $(\mathbb{Z}_n, +_n)$
  - 3 is a primitive root of $(\mathbb{Z}^*_7, \times_7)$
- Not all groups do have primitive roots and those who have are called *cyclic groups*
- <u>Theorem 5:</u>
  $(\mathbb{Z}^*_n, \times_n)$ does have a primitive root $\Leftrightarrow n \in \{2, 4, p, 2 \times p^e\}$ where p is an odd prime and $e \in \mathbb{Z}^+$
  - For the proof see [Niv80a]

Some More Mathematical Background (5)

- Theorem 6:

  If $(S, \bullet)$ is a group and $b \in S$ then $(S', \bullet)$ with $S' = \{b^a \mid a \in \mathbb{Z}^+\}$ is also a group.
  - For the proof refer to [Cor90a] section 33.3
  - As $S' \subseteq S$, $(S', \bullet)$ is called a *subgroup* of $(S, \bullet)$
  - If b is a primitive root of $(S, \bullet)$ then $S' = S$
- Definition: *order of a group and of an element*
  - Let $(S, \bullet)$ be a group, $e \in S$ its identity element and $b \in S$ any element of S:
    - Then $|S|$ is called the *order* of $(S, \bullet)$
    - Let $c \in \mathbb{Z}^+$ be the smallest element so that $b^c = e$ (if such a c exists, if not set $c = \infty$). Then c is called the *order* of b.

Some More Mathematical Background (6)

- Theorem 7 (Lagrange):

  If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$. Hence, if $b \in G$ then the order of b divides $|G|$.
- Theorem 8:

  If G is a cyclic finite group of order n and d divides n then G has exactly $\Phi(d)$ elements of order d. In particular, G has $\Phi(n)$ elements of order n.
- Theorems 5, 7, and 8 are the basis of the following algorithm that finds a cyclic group $\mathbb{Z}^*_p$ and a primitive root g of it:
  - Choose a large prime q such that p = 2q + 1 is prime.
    - As p is prime, Theorem 5 states that $\mathbb{Z}^*_p$ is cyclic.
    - The order of $\mathbb{Z}^*_p$ is $2 \times q$ and $\Phi(2 \times q) = \Phi(2) \times \Phi(q) = q - 1$ as q is prime.
    - So, the odds of randomly choosing a primitive root are $(q - 1) / 2q \approx 1 / 2$
    - In order to efficiently test, if a randomly chosen g is a primitive root, we just have to test if $g^2 \equiv 1 \bmod p$ or $g^q \equiv 1 \bmod p$. If not, then its order has to be $|\mathbb{Z}^*_p|$, as Theorem 7 states that the order of g has to divide $|\mathbb{Z}^*_p|$

Some More Mathematical Background (7)

- Definition: *discrete logarithm*
  - Let p be prime, g be a primitive root of $(\mathbb{Z}^*_p, \times_p)$ and c be any element of $\mathbb{Z}^*_p$. Then there exists z such that: $g^z \equiv c \bmod p$

    z is called the *discrete logarithm* of c modulo p to the base g
  - Example 6 is the discrete logarithm of 1 modulo 7 to the base 3 as $3^6 \equiv 1 \bmod 7$
  - The calculation of the discrete logarithm z when given g, c, and p is a computationally difficult problem and the asymptotical runtime of the best known algorithms for this problem is exponential in the bitlength of p

Diffie-Hellman Key Exchange (2)

- If Alice (A) and Bob (B) want to agree on a shared secret s and their only means of communication is a public channel, they can proceed as follows:
  - A chooses a prime p, a primitive root g of $\mathbb{Z}^*_p$, and a random number q:
    - A and B can agree upon the values p and g prior to any communication, or A can choose p and g and send them with his first message
    - A computes $v = g^q$ MOD p and sends to B: $\{p, g, v\}$
  - B chooses a random number r:
    - B computes $w = g^r$ MOD p and sends to A: $\{p, g, w\}$ (or just $\{w\}$)
  - Both sides compute the common secret:
    - A computes $s = w^q$ MOD p
    - B computes $s' = v^r$ MOD p
    - As $g^{(q \times r)}$ MOD $p = g^{(r \times q)}$ MOD p it holds: $s = s'$
  - An attacker Eve who is listening to the public channel can only compute the secret s, if she is able to compute either q or r which are the discrete logarithms of v, w modulo p to the base g

TELEMATIK Rechnernetze

Diffie-Hellman Key Exchange (3)

- If the attacker Eve is able to alter messages on the public channel, she can launch a *man-in-the-middle attack:*
  - Eve generates to random numbers q' and r':
    - Eve computes $v' = g^{q'}$ MOD p and $w' = g^{r'}$ MOD p
  - When A sends $\{p, g, v\}$ she intercepts the message and sends to B: $\{p, g, v'\}$
  - When B sends $\{p, g, w\}$ she intercepts the message and sends to A: $\{p, g, w'\}$
  - When the supposed "shared secret" is computed we get:
    - A computes $s_1 = w'^q$ MOD p = $v^{r'}$ MOD p the latter computed by E
    - B computes $s_2 = v'^r$ MOD p = $w^{q'}$ MOD p the latter computed by E
    - So, in fact A and E have agreed upon a shared secret s_1 as well as E and B have agreed upon a shared secret s_2
  - If the "shared secret" is now used by A and B to encrypt messages to be exchanged over the public channel, E can intercept all the messages and decrypt / re-encrypt them before forwarding them between A and B.

TELEMATIK Rechnernetze

Diffie-Hellman Key Exchange (4)

- Two countermeasures against the man-in-the-middle attack:
  - The shared secret is *"authenticated"* after it has been agreed upon
    - We will treat this in the section on key management
  - A and B use a so-called *interlock protocol* after agreeing on a shared secret:
    - For this they have to exchange messages that E has to relay before she can decrypt / re-encrypt them
    - The content of these messages has to be checkable by A and B
    - This forces E to invent messages and she can be detected
    - One technique to prevent E from decrypting the messages is to split them into two parts and to send the second part before the first one.
      - If the encryption algorithm used inhibits certain characteristics E can not encrypt the second part before she receives the first one.
      - As A will only send the first part after he received an answer (the second part of it) from B, E is forced to invent two messages, before she can get the first parts.
- Remark: In practice the number g does not necessarily need to be a primitive root of p, it is sufficient if it generates a large subgroup of $\mathbb{Z}^*_p$

© Dr.-Ing G. Schäfer

The ElGamal Algorithm (1)

- The ElGamal algorithm can be used for both, encryption and digital signatures (see also [ElG85a])
- Like the DH exchange it is based on the difficulty of computing discrete logarithms in finite fields
- In order to set up a key pair:
  - Choose a large prime p, a generator g of the multiplicative group $\mathbb{Z}^*_p$ and a random number v such that $1 \leq v \leq p - 2$. Calculate: $y = g^v \bmod p$
  - The public key is (y, g, p)
  - The private key is v
- To sign a message m:
  - Choose a random number k such that k is relatively prime to $p - 1$.
  - Compute $r = g^k \bmod p$
  - With the Extended Euclidean Algorithm compute k^{-1}, the inverse of $k \bmod (p - 1)$
  - *Compute* $s = k^{-1} \times (m - v \times r) \bmod (p - 1)$
  - The signature over the message is (r, s)

The ElGamal Algorithm (2)

- To verify a signature (r, s) over a message m:
  - Confirm that $y^r \times r^s \text{ MOD } p = g^m \text{ MOD } p$
  - Proof: We need the following
    - Lemma 3:
      Let p be prime and g be a generator of $\mathbb{Z}^*_p$.
      Then $i \equiv j \bmod (p - 1) \Rightarrow g^i \equiv g^j \bmod p$
      Proof:
      – $i \equiv j \bmod (p - 1) \Rightarrow$ there exists $k \in \mathbb{Z}^+$ such that $(i - j) = (p - 1) \times k$
      – So, $g^{(i-j)} = g^{(p-1) \times k} \equiv 1^k \equiv 1 \bmod p$, because of Theorem 3 (Euler)
      $\Rightarrow g^i \equiv g^j \bmod p$
    - So as

$$
\begin{array}{lll}
s \equiv k^{-1} \times (m - v \times r) & \bmod (p - 1) & \\
\Leftrightarrow \quad k \times s \equiv m - v \times r & \bmod (p - 1) & \\
\Leftrightarrow \quad m \equiv v \times r + k \times s & \bmod (p - 1) & \\
\Rightarrow \quad g^m \equiv g^{(v \times r + k \times s)} & \bmod p & \text{with Lemma 3} \\
\Leftrightarrow \quad g^m \equiv g^{(v \times r)} \times g^{(k \times s)} & \bmod p & \\
\Leftrightarrow \quad g^m \equiv y^r \times r^s \quad \bmod p & &
\end{array}
$$

The ElGamal Algorithm (3)

- Security of ElGamal signatures:
  - As the private key *v* is needed to be able to compute *s*, an attacker would have to compute the discrete logarithm of *y* modulo *p* to the basis *g* in order to forge signatures
  - It is crucial to the security, that a new random number *k* is chosen for every message, because an attacker can compute the secret *v* if he gets two messages together with their signatures based on the same *k* (see [Men97a], Note 11.66.ii)
  - In order to prevent an attacker to be able to create a message M with a matching signature, it is necessary not to sign directly the message M as explained before, but to sign a cryptographic hash value m = h(M) of it (these will be treated soon, see also [Men97a], Note 11.66.iii)

The ElGamal Algorithm (4)

- To encrypt a message *m* using the public key (*y, g, p*):
  - Choose a random $k \in \mathbb{Z}^+$ with $k < p - 1$
  - Compute $r = g^k \text{ MOD } p$
  - Compute $s = m \times y^k \text{ MOD } p$
  - The ciphertext is (*r*, *s*), which is twice as long as *m*
- To decrypt the message (*r, s*) using *v:*
  - Use the private key *v* to compute $r^{(p-1-v)} \text{ MOD } p = r^{(-v)} \text{ MOD p}$
  - Recover *m* by computing $m = r^{(-v)} \times s \text{ MOD } p$
  - Proof:

$$r^{(-v)} \times s \equiv r^{(-v)} \times m \times y^k \equiv g^{(-vk)} \times m \times y^k \equiv g^{(-v \times k)} \times m \times g^{(v \times k)} \equiv m \text{ mod } p \quad \blacksquare$$

- Security:
  - The only known means for an attacker to recover *m* is to compute the discrete logarithm *v* of *y* modulo *p* to the basis *g*
  - For every message a new random *k* is needed ([Men97a], Note 8.23.ii)

Elliptic Curve Cryptography (1)

- The algorithms presented so far have been invented for the multiplicative group ($\mathbb{Z}^*_p$, $\times_p$) and the field ($\mathbb{Z}_p$, $+_p$, $\times_p$), respectively
- It has been found during the 1980's that they can be generalized and be used with other groups and fields as well
- The main motivation for this generalization is:
  - A lot of mathematical research in the area of primality testing, factorization and computation of discrete logarithms has led to techniques that allow to solve these problems in a more efficient way, if certain properties are met:
    - When the RSA-129 challenge was given in 1977 it was expected that it will take some 40 quadrillion years to factor the 129-digit number ($\approx$ 428 bit)
    - In 1994 it took 8 months to factor it by a group of computers networked over the Internet, calculating for about 5000 MIPS-years
    - Advances in factoring algorithms allowed 2009 to factor a 232-digit number (768 bit) in about 1500 AMD64-years [KAFL10]

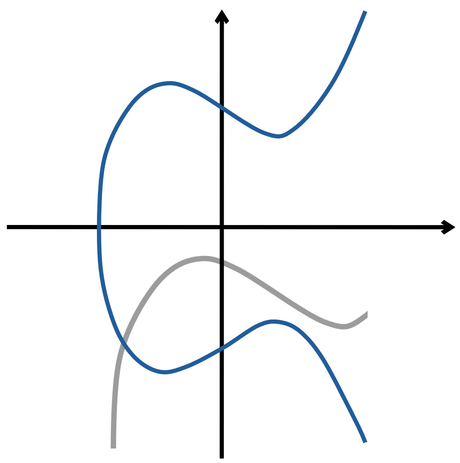
    $\Rightarrow$ the key length has to be increased (currently about 2048 bit)

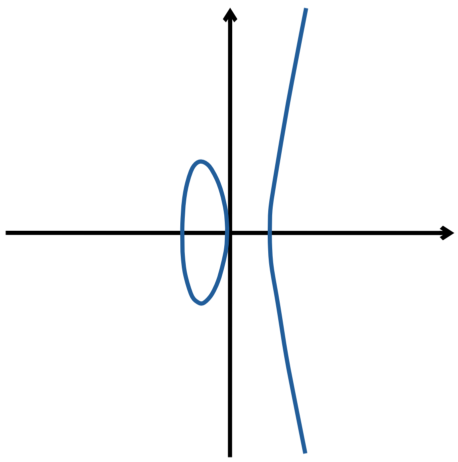
Elliptic Curve Cryptography (2)

- Motivation (continued):
  - Some of the more efficient techniques do rely on specific properties of the algebraic structures ($\mathbb{Z}^*_p$, $\times_p$) and ($\mathbb{Z}_p$, $+_p$, $\times_p$)
  - Different algebraic structures may therefore provide the same security with shorter key lengths
- A very promising structure for cryptography can be obtained from the *group of points on an elliptic curve over a finite field*
  - The mathematical operations in these groups can be efficiently implemented both in hardware and software
  - The discrete logarithm problem is believed to be hard in the general class obtained from the group of points on an elliptic curve over a finite field

Foundations of ECC - Group Elements

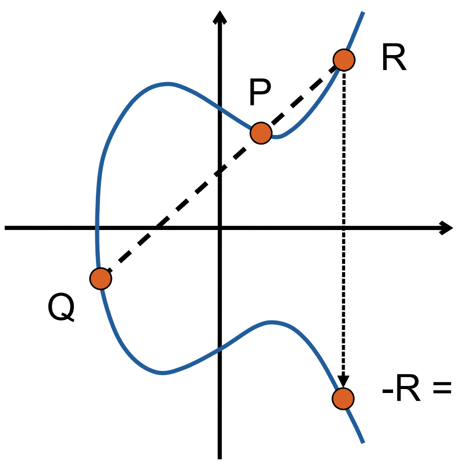
- Algebraic group consisting of
  - Points on Weierstrass' Equation: $y^2 = x^3 + ax + b$
  - Additional point O in "infinity"
- May be calculated over $\mathbb{R}$, but in cryptography $\mathbb{Z}_p$ and $GF(2^n)$ are used
- Already in $\mathbb{R}$ arguments influence form significantly:
  - $y^2 = x^3 - 3x + 5$ $y^2 = x^3 - 40x + 5$





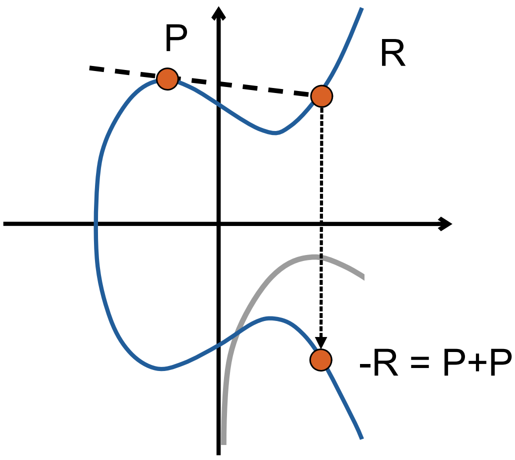
Foundations of ECC - Point Addition

- Addition of elements = Addition of points on the curve
- Geometric interpretation:
  - Each point P: (x,y) has an inverse -P: (x,-y)
  - A line through two points P and Q usually intersects with a third point R
  - Generally, sum of two points P and Q equals –R



Foundations of ECC - Point Addition (Special cases)

- The additional point O is the neutral element, i.e., P + O = P
- P + (-P):
  - If the inverse point is added to P, the line and curve intersect in “infinity”
  - By definition: P + (-P) = O
- P + P: The sum of two identical points P is the inverse of the intersecting point with the tangent through P:



Foundations of ECC - Algebraic Addition

- If one of the summands is O, the sum is the other summand
- If the summands are inverse to each other the sum is O
- For the more general cases the slope of the line is:

$$\alpha = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{for } P \neq -Q \wedge P \neq Q \\ \frac{3x_P^2 + a}{2y_P} & \text{for } P = Q \end{cases}$$

- Result of point addition, where (x_r, y_r) is already the reflected point (-R)

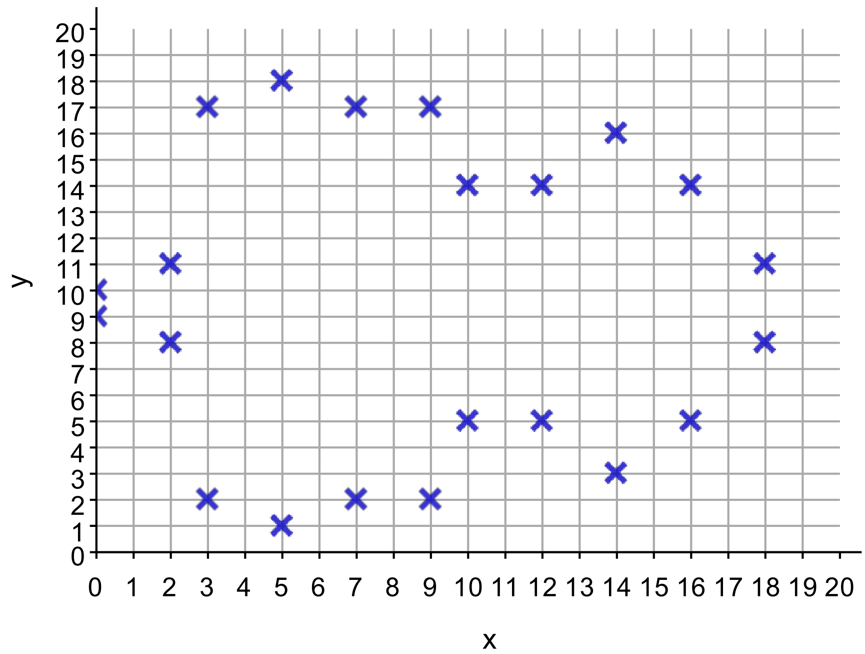
$$x_r = \alpha^2 - x_p - x_q$$
$$y_r = \alpha(x_p - x_r) - y_p$$

Foundations of ECC - Multiplication

- Multiplication of natural number *n* and point *P* performed by multiple repeated additions
- Numbers are grouped into powers of 2 to achieve logarithmic runtime, e.g. 25P = P + 8P + 16P
- This is possible if and only if the n is known!
- If n is unknown for nP = Q, a logarithm has to be solved, which is possible if the coordinate values are chosen from $\mathbb{R}$
- For $\mathbb{Z}_p$ and GF(2^n) the discrete logarithm problem for elliptic curves has to be solved, which cannot be done efficiently!
- *Note:* it is not defined how two points are multiplied, but only a natural number *n* and point *P*

Foundations of ECC – Curves over $\mathbb{Z}_p$

- Over $\mathbb{Z}_p$ the curve degrades to a set of points
- For $y^2 \equiv x^3 - 3x + 5 \mod 19$:



- Note: For some x values, there is no y value!

Foundations of ECC – Calculate the y-values in $\mathbb{Z}_p$

- In general a little bit more problematic: determine the y-values for a given x (as its square value is calculated) by $y^2 \equiv f(x) \bmod p$
- Hence p is often chosen s.t. $p \equiv 3 \bmod 4$
- Then y is calculated by $y_1 \equiv f(x)^{\frac{p+1}{4}} \bmod p$ and $y_2 \equiv -f(x)^{\frac{p+1}{4}} \bmod p$ if and only if a solution exists at all
- Short proof:
  - From the Euler Theorem 3 we know that $f(x)^{p-1} \equiv 1 \bmod p$
  - Thus the square root must be 1 or -1 $f(x)^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$
  - Case 1: $f(x)^{\frac{p-1}{2}} \equiv 1 \bmod p$
    - Multiply both sides by f(x): $f(x)^{\frac{p+1}{2}} \equiv f(x) \equiv y^2 \bmod p$
    - As p + 1 is divisible by 4 we can take the square root so that $f(x)^{\frac{p+1}{4}} \equiv y \bmod p$ ■
  - Case 2: In this case no solution exists for the given x value (as shown by Euler)

Foundations of ECC – Addition and Multiplication in $\mathbb{Z}_p$

- Due to the discrete structure point mathematical operations do not have a geometric interpretation any more, but
- Algebraic addition similar to addition over $\mathbb{R}$
- If the inverse point is added to P, the line and “curve” still intersect in “infinity”
- All x- and y-values are calculated mod p
- Division is replaced by multiplication with the inverse element of the denominator
    - Use the Extended Euclidean Algorithm with *w* and *p* to derive the inverse -*w*
- Algebraic multiplication of a natural number *n* and a point *P* is also performed by repeated addition of summands of the power of 2
- The discrete logarithm problem is to determine a natural number *n* in *nP* = Q for two known points *P* and *Q*

Foundations of ECC – Size of generated groups

- Please note that the order of a group generated by a point on a curve over $\mathbb{Z}_p$ is not p-1!
- Determining the exact order is not easy, but can be done in logarithmic time by Schoofs algorithm [Sch85] (requires much more mathematical background than desired here)
- But Hasse's theorem on elliptic curves states that the group size n must lay between:
- $p + 1 - 2\sqrt{p} \leq n \leq p + 1 + 2\sqrt{p}$
- As mentioned before: Generating rather large groups is sufficient

Foundations of ECC - ECDH

- The Diffie-Hellman-Algorithm can easily be adapted to elliptic curves
- If Alice (A) and Bob (B) want to agree on a shared secret *s*:
  - A and B agree on a cryptographically secure elliptic curve and a point *P* on that curve
  - A chooses a random number *q:*
    - A computes $Q = q\ P$ and transmits *Q* to Bob
  - B chooses a random number *r:*
    - B computes $R = r\ P$ and transmits *P* to Alice
  - Both sides compute the common secret:
    - A computes $S\ \ = q\ R$
    - B computes $S' = r\ Q$
    - As $q\ r\ P = r\ q\ P$ the secret point $S = S'$
- Attackers listening to the public channel can only compute S, if able to compute either *q* or *r* which are the discrete logarithms of Q and *R* for the point *P*

Foundations of ECC – EC version of ElGamal Algorithm (I)

- Adapting ElGamal for elliptic curves is rather straight forward for the encryption routine
- To set up a key pair:
  - Choose an elliptic curve over a finite field, a point G that generates a large group, and a random number v such that $1 < v < n$, where n denotes to the size of the induced group, Calculate: $Y = vG$
  - The public key is (Y, G, *curve*)
  - The private key is v

TELEMATIK Rechnernetze

Foundations of ECC – EC version of ElGamal Algorithm (II)

- To encrypt a message:
  - Choose a random $k \in \mathbb{Z}^+$ with $k < n - 1$, compute $R = kG$
  - Compute $S = M + kY$, where M is a point derived by the message
    - Problem: Interpreting the message m as a x coordinate of M is not sufficient, as the y value does not have to exist
    - Solution from [Ko87]: Choose a constant c (e.g. 100) check if cm is the x coordinate of a valid point, if not try $cm+1$, then $cm+2$ and so on
    - To decode m: take the x value of M and do an integer division by c (receiver has to know c too)
  - The ciphertext are the points (R, S)
  - Twice as long as m, if stored in so-called *compressed form*, i.e. only x coordinates are stored and a single bit, indicating whether the larger or smaller corresponding y-coordinate shall be used
- To decrypt a message:
  - Derive M by calculating $S - vR$
  - Proof: $S - vR = M + kY - vR = M + kvG - vkG = M + O = M$

Foundations of ECC – EC version of ElGamal Algorithm (II)

- To sign a message:
  - Choose a random $k \in \mathbb{Z}^+$ with $k < n - 1$, compute $R = kG$
  - Compute $s = k^{-1}(m + rv) \bmod n$, where r is the x-value of R
  - The signature are (r, s), again about as twice as long as n
- To verify a signed message:
  - Check if the *point* $P = ms^{-1}G+rs^{-1}Y$ has the x-coordinate r
  - *Note*: s^{-1} is calculated by the Extended Euclidian Algorithm with the input s and n (the order of the group)
  - Proof: $ms^{-1}G+rs^{-1}Y = ms^{-1}G+rs^{-1}vG = (m+rv)(s^{-1})G = (ks)(s^{-1})G = kG = R$

- Security discussion:
  - As in the original version of ElGamal it is crucial to not use k twice
  - Messages should not be signed directly
  - Further checks may be required, i.e., G must not be O, a valid point on the curve etc. (see [NIST09] for further details)

Foundations of ECC – Security (I)

- The security heavily depends on the chosen curve and point:
- The discriminant of the curve must not be zero, i.e., $4a^3 + 27b^2 \not\equiv 0 \bmod p$ otherwise the curve is degraded (a so called *singular curve*)
- Menezes et. al. have found a sub-exponential algorithm for so-called *supersingular elliptic curves* but this does not work in the general case [Men93a]
- The constructed algebraic groups should have as many elements a possible
- This class will not go into more details of elliptic curve cryptography as this requires way more mathematics than desired for this course... :o)
- For non-cryptographers it is best to depend on predefined curves, e.g., [LM10] or [NIST99] and standards such as ECDSA
- Many publications choose parameters a and b such that they are provably chosen by a random process (e.g. publish x for h(x) = a and y for h(y) = b); Shall ensure that the curves do not contain a cryptographic weakness that only the authors knows about

Foundations of ECC – Security (II)

- The security depends on the length of *p*
  - Key lengths with comparable strengths according to [NIST12]:

Symmetric Algorithms	RSA	ECC
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	> 512

Foundations of ECC – Security (III)

- The security also heavily depends on the implementation!
  - The different cases (e.g. with O) in ECC calculation may be observable, i.e., power consumption and timing differences
  - Attackers might deduct side-channel attacks, as in OpenSSL 0.9.8o [BT11]
    - Attacker may deduce the bit length of a value k in kP by measuring the time required for the square and multiply algorithm
    - Algorithm was aborted early in OpenSSL when no further bits where set to "1"
  - Attackers might try to generate invalid points to derive facts about the used key as in OpenSSL 0.9.8g, leading to a recovery of a full 256-bit ECC key after only 633 queries [BBP12]
- *Lesson learned:* Do not do it on your own, unless you have to and know what you are doing!

Foundations of ECC – Further remarks

- As mentioned earlier it is possible to construct cryptographic elliptic curves over $G(2^n)$, which may be faster in hardware implementations
  - We refrained from details as this would not have brought many different insights!
- Elliptic curves and similar algebraic groups are an active field of research and allow other advanced applications e.g.:
  - So-called Edwards Curves are currently discussed, as they seem more robust against side-channel attacks (e.g. [BLR08])
  - Bilinear pairings allow
    - Programs to verify that they belong to the same group, without revealing their identity (Secret handshakes, e.g. [SM09])
    - Public keys to be structured, e.g. use “Alice” as public key for Alice (Identity based encryption, foundations in [BF03])
- Before deploying elliptic curve cryptography in a product, make sure to not violate patents, as there are still many valid ones in this field!

Conclusion

- Asymmetric cryptography allows to use two different keys for:
  - Encryption / Decryption
  - Signing / Verifying
- The most practical algorithms that are still considered to be secure are:
  - RSA, based on the difficulty of factoring and solving discrete logarithms
  - Diffie-Hellman (not an asymmetric algorithm, but a key agreement protocol)
  - ElGamal, like DH based on the difficulty of computing discrete logarithms
- As their security is entirely based on the difficulty of certain mathematical problems, algorithmic advances constitute their biggest threat
- Practical considerations:
  - Asymmetric cryptographic operations are about magnitudes slower than symmetric ones
  - Therefore, they are often not used for encrypting / signing bulk data
  - Symmetric techniques are used to encrypt / compute a cryptographic hash value and asymmetric cryptography is just used to encrypt a key / hash value

Additional References

[Bre88a] D. M. Bressoud. *Factorization and Primality Testing.* Springer, 1988.

[Cor90a] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[DH76] W. Diffie, M. E. Hellman. *New Directions in Cryptography.* IEEE Transactions on Information Theory, IT-22 , pp. 644-654, 1976.

[ElG85a] T. ElGamal. *A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.* IEEE Transactions on Information Theory, Vol.31, Nr.4, pp. 469-472, July 1985.

[Kob87a] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer, 1987.

[Men93a] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.

[Niv80a] I. Niven, H. Zuckerman. *An Introduction to the Theory of Numbers.* John Wiley & Sons, 4th edition, 1980.

[RSA78] R. Rivest, A. Shamir und L. Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.* Communications of the ACM, February 1978.

Additional References

[KAFL10] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. Osvik, H. Te Riele, A.Timofeev, P. Zimmermann. *Factorization of a 768-bit RSA modulus*. In Proceedings of the 30th annual conference on Advances in cryptology (CRYPTO'10), 2010.

[LM10] M. Lochter, J. Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*, IETF Request for Comments: 5639, 2010.

[NIST99] NIST. *Recommended Elliptic Curves for Federal Government Use*. 1999.

[NIST12] NIST. *Recommendation for Key Management: Part 1: General (Revision 3).* NIST Special Publication 800-57. 2012.

[Ko87] N. Koblitz. *Elliptic Curve Cryptosystems.* Mathematics of Computation, Vol. 48, No. 177 (Jan., 1987), pp. 203-209. 1987.

[BBP12] B.B. Brumley, M. Barbosa, D. Page, F. Vercauteren*. Practical realisation and elimination of an ECC-related software bug attack.* Cryptology ePrint Archive: Report 2011/633 and CT-RSA Pages 171-186. 2012.

[BT11] B.B. Brumley, N. Tuveri*. Remote timing attacks are still practical.* Proceedings of the 16th European conference on Research in computer security (ESORICS'11). Pages 355-371. 2011.

Additional References

[BLR08] D. Bernstein, T. Lange, R. Rezaeian Farashahi. *Binary Edwards Curves.* Cryptographic Hardware and Embedded Systems (CHES). Pages 244-265. 2008.

[NIST09] NIST. *Digital Signature Standard (DSS).* FIPS PUB 186-3. 2009.

[SM09] A. Sorniotti, R. Molva. *A provably secure secret handshake with dynamic controlled matching.* Computers & Security, 2009.

[BF03] D. Boneh, M. Franklin. *Identity-Based Encryption from the Weil Pairing.* SIAM J. of Computing, Vol. 32, No. 3, Pages 586-615, 2003.

[Sch85] R. Schoof. *Elliptic Curves over Finite Fields and the Computation of Square Roots mod p*. Math. Comp., 44(170). Pages 483–494. 1985.