Addition to last lecture:

I have just explained for the index-calculus-alg. how to calculate „$\log_\alpha(p_i)$" of the factor base $(p_i)_i$, but not how to calculate the discrete log $\log_\alpha(\beta)$

Therefore, on page 2 of the last lecture notes, before „• Most efficiently" it needs to be added:

- Take random b, until $\alpha^b \cdot \beta = \prod_{i=1}^{t} p_i^{\lambda_i}$ can be found

$\Rightarrow$ $\cancel{b+}$ $\log_\alpha(\beta) = \sum_i \lambda_i \log_\alpha(p_i) - b$

Remarks to probabilistic procedure of determining point on EC to a message M.

- Obviously the procedure returns a point on an EC with high prob.
- In $\mathbb{F}_p$ half of the elements are quadratic residues. Assuming that you may pick any quadratic residue with probability $1/2$ (even if they are neighbours), the probability of failing to find one in the above alg. is given by $\left(\frac{1}{2}\right)^{2^k}$
- If a point (x, y) on the EC is given, the corresponding original integer message will be $M = \left\lfloor \frac{x}{2^k} \right\rfloor$.

Analogously to the deterministic approach, the message space needs to be reduced such that unique demapping is possible.

13.4.3 ElGamal on elliptic curves (EC ElGamal)

A's private key : random number $x_a \in \{2, \ldots, n-2\}$

A's public key : $x_a \cdot P$

B wants to encrypt $m \in \langle P \rangle$, ~~therefore~~ for getting m we may apply 13.4.2

(i) B chooses random $k \in \{2, \ldots, n-2\}$, computes $Q = k \cdot P$

(ii) B computes $R = k(x_a \cdot P) + m$

(iii) $B \to A$: (Q, R)

A deciphers :

(i) A computes $x_a \cdot Q = x_a \cdot k \cdot P$

(ii) A computes $R - x_a \cdot Q = \cancel{k(x_a \cdot P)} + m - \cancel{x_a k \cdot P} = m$

Use demapping of 13.4.2

13.4.4 Elliptic Curve Integrated Encryption Scheme (ECIES)

ECIES is a variant of ElGamal introduced by Bellare, Rogaway.
A Diffie-Hellman shared secret is used to derive two symmetric keys K_1 and K_2. K_1 is used for symmetric encryption and K_2 for ciphertext authentication.

We need the following primitives:

- Symmetric encryption function: ENC_{K_1} (e.g., AES)
  with corresponding decryption: DEC_{K_1}
- Message authentication code: MAC_{K_2} (e.g., HMAC)
- key derivation function: KDF
  $(K_1, K_2) = KDF(S) = H(S,0) \,||\, H(S,1) \,||\, H(S,2) \ldots \,||\, H(S,i) \,||\ldots$
  until enough bits for K_1 and K_2 are generated. H is hash function

System parameters:

$\mathbb{F}_p$, p prime, $E: y^2 = x^3 + ax + b \,/\, \mathbb{F}_p$, $P \in E(\mathbb{F}_p)$
s.t. $\mathrm{ord}(P) = n$ prime, $h = \dfrac{\# E(\mathbb{F}_p)}{n}$

Key generation

- Choose a random $d \in \{2, \ldots, n-2\}$ (private key)
- Compute $Q = dP$ (public key)

Encryption of $m \in \{0,1\}^*$

1) Choose a random $k \in \{2, \ldots, n-2\}$
2) $R \leftarrow k \cdot P$, $Z \leftarrow h \cdot k \cdot Q$, if $Z = O$ goto 1.)
3) $(K_1, K_2) = KDF(x_Z, R)$ where x_Z is the x-coordinate of Z
4) $C \leftarrow ENC_{K_1}(m)$ $t \leftarrow MAC_{K_2}(C)$
5) Send (R, C, t)

Decryption

1) Assume the validity of R : - $R \neq O$
   - $R \in E(\mathbb{F}_p)$
2) $Z \leftarrow h d R$, check whether $Z \neq O$
3) $(k_1, k_2) = KDF(x_Z, R)$
4) $t' \leftarrow MAC_{k_2}(c)$, check whether $t' = t$
5) $m \leftarrow DEC_{k_1}(c)$

To show that decryption works, only prove that Z is computed correctly:

$h \cdot d \cdot R = h \cdot d \cdot k \cdot P = h \cdot k \cdot d \cdot P = h \cdot k \cdot Q$ ✓

13.4.5 Elliptic Curve Digital Signature Algorithm (ECDSA)

EC : $E/\mathbb{F}_p$, generator P, $ord(P) = n$, with $L_n = \lceil \log_2(n) \rceil$ and a hash function h.

x: private key $1 < x < n-1$

$Y = x \cdot P$ is the public key

Alg 14 Creating signature (r, s) on a message m

Input: Message m
Output: A signature (r, s) on m

$e \Leftarrow h(m)$
$z \Leftarrow$ the L_n leftmost bits of e
Repeat
  Repeat
    Select random $1 \leq k \leq n-1$ with $gcd(k, n) = 1$
    $(x_1, y_1) \Leftarrow kP$
    $r \Leftarrow x_1 \bmod n$
  Until $r \neq 0$
  $s \Leftarrow k^{-1}(z + x \cdot r) \bmod n$
Until $gcd(s, n) = 1$
Return (r, s)

Alg 15 Verifying signature (r, s) on a message m

Input: Message m and (r, s)
Output: Acceptance or denial of signature (r, s) on message m

Verify that $1 \leq r, s \leq n-1$
$e \Leftarrow h(m)$
$z \Leftarrow$ the L_n leftmost bits of e
$w \Leftarrow s^{-1} \bmod n$
$u_1 \Leftarrow wz \bmod n$
$u_2 \Leftarrow rw \bmod n$
$(v, y_1) \Leftarrow u_1 \cdot P + u_2 \cdot Y$
if $v \bmod n = r$ then
  return accept
else
  return deny
endif

The verification process holds true as

$$(r, y_1) = u_1 \cdot P + u_2 \cdot Y = s^{-1} z \cdot P + r \cdot s^{-1} \cdot x \cdot P$$

$$= s^{-1}(z + r \cdot x) \cdot P = k \cdot P = (x_1, y_1) \equiv (r, y_1) \mod n$$