Identity-Based Cryptography

T-79.5502 Advanced Course in Cryptology

Billy Brumley

`billy.brumley at hut.fi`

Laboratory for Theoretical Computer Science

Helsinki University of Technology

Outline

- Classical ID-Based Crypto; Self-Certified Keys
- Bilinear Maps, ID-Based schemes
- Elliptic Curves
- Tate Pairing, implementation

Identity-Based Cryptography

- PKI doesn't really scale well. Instead (Shamir 84), let's use a user's *Identity (ID)* as a public key.
- $\Rightarrow$ key channels are no longer needed! (don't have to "lookup" a user's public key)
- *Unconditional Trust* of the *Trusted Third Party (TTP)*. (Well, most of the time...)
- $\Rightarrow$ TTP can **read** everything and **forge** everything. Acceptable in some cases?

Shamir's ID-Based Signatures

Algorithm 13.1:

- **Setup.** (TTP) $N = pq$, $e \in \mathbb{Z}_N^*$, $d = e^{-1} \mod \phi(N)$, $\mathsf{H} : \{0,1\}^* \mapsto \mathbb{Z}_{\phi(N)}$. $\{N, e, \mathsf{H}\}$ are public, system-wide parameters; d is TTP's private *master key*.
- **User Keygen.** $g = ID^d \pmod N$ (g is private)
- **Sign.** $t = r^e \pmod N$, $s = g \cdot r^{\mathsf{H}(t\|M)} \pmod N$ where $r \in_R \mathbb{Z}_N^*$ and M is the message; the signature is $\{s, t\}$.
- **Verify.** TRUE iff $s^e \equiv ID \cdot t^{\mathsf{H}(t\|M)} \pmod N$

$$s^e = (gr^{\mathsf{H}(t\|M)})^e = ID^{de} r^{e \cdot \mathsf{H}(t\|M)} = ID \cdot t^{\mathsf{H}(t\|M)}$$

Self-Certified Keys

- Instead of verifying public keys from a TTP signature, *extract* the public key using the user's *identity* (ID) (Girault 91).
- *Reduced storage*: a TTP signature on a key is no longer needed.
- *Computationally efficient*: keys can be extracted using only 1 exponentation (scalar mult), while verifying public keys from a signature takes 2.
- The authenticity of SC keys cannot be explicitely verified. (The authenticity is *implicit*, so they are sometimes called **implicit certificates**.)
- SC keys can only be used with the same cryptographic settings in which they were generated.

Implicit Certificates

TTP has private key v and public key V. Function f maps a group element R and message m to $\mathbb{Z}_q$ as $f(R,m) \mapsto \mathsf{P}(R) + \mathsf{H}(m) \mod q$ where P is a projection and H a hash function. (It is public, anyone can calculate it.)

TTP $\longleftarrow$ Alice: G^a

$$\text{TTP: } \begin{cases} R = & G^k G^a \\ r = & f(R, ID) \\ \bar{s} = & -k - vr \pmod{q} \end{cases}$$

Alice $\longleftarrow$ TTP: $\{R, \bar{s}\}$

Alice's private key is $s = a - \bar{s} \pmod{q}$. The implicit certificate is R and Alice's public key is extracted by first computing $r = f(R, ID)$ then

$$V^r R = G^{vr+k+a} = G^{vr+k+\bar{s}+s} = G^{vr+k-k-vr+s} = G^s$$

Properties of Bilinear Maps

- *Exponentation in groups*. Denoted $g^k = g \cdot g \cdot \ldots \cdot g$ in $(G, \cdot)$ or $kP = P + P + \cdots + P$ in $(G, +)$. (Both k times.)
- Consider the two groups $(G_1, +)$ and $(G_2, \cdot)$ of prime order q. A **bilinear** map

$$e : G_1 \times G_1 \to G_2$$

  has three useful properties:
  - **Bilinearity.** $\forall P, Q \in G_1, \forall a, b \in \mathbb{Z}_q^*$, $e(aP, bQ) = e(bP, aQ) = e(P, Q)^{ab}$
  - **Non-Degeneracy.** $\forall P \in G_1 \setminus O, e(P, P) \neq 1$. (Hence $e(P, P)$ generates G_2.)
  - **Computability.** e is efficiently computable.
- Typically, G_1 is an elliptic curve and G_2 a finite field. (The notation reflects this.)

Bilinear Maps & Discrete Logs

Theorem 1. *The Discrete Log Problem in G_1 is no harder than the Discrete Log Problem in G_2.*

Proof. Given $Q = aP \in G_1$, we want to know $\log_P Q$ in G_1. From bilinearity, we have $e(P,Q) = e(P,aP) = e(P,P)^a$ so we calculate $P' = e(P,P) \in G_2$ and $Q' = e(P,Q) \in G_2$. We then calculate $a = \log_{P'} Q'$ in G_2, and $a = \log_P Q$ in G_1 also holds. $\square$

Decisional DH Problem (DDH)

Definition 13.1.
Decisional Diffie-Hellman (DDH) Problem: in $(G, +)$:

- **INPUT:** Four elements $P, aP, bP, cP \in G$. P generates G.
- **OUTPUT:** YES iff $c \equiv ab \pmod{\#G}$.

DDH can't be harder than CDH; given a CDH solver, one can solve DDH.

Can DDH be easy if CDH is hard?

Bilinear Maps & Decisional DH

Theorem 2. *The Decisional Diffie-Hellman Problem is easy in* G_1.

Proof. Given $P, aP, bP, cP \in G_1$ with $a, b, c \in_R \mathbb{Z}_q^*$, it follows that

$$e(aP, bP) = e(P, P)^{ab} \text{ and}$$
$$e(P, cP) = e(P, P)^{c}$$

As e is non-degenerate, $c \equiv ab \pmod{q}$ iff $e(aP, bP) = e(P, cP)$. □

1-Round 3-Party DH Key Agreement

- **Joux 00, Sec. 13.3.6 Mao.** *Not ID-Based*. One-round tripartite DH key agreement; classical DH takes more rounds. *Wonderfully simple!*
- Assume the previous notation with $e : G_1 \times G_1 \to G_2$ a bilinear map and P a generator of G_1 with order q.
- Three parties A, B, C have private keys $a, b, c \in \mathbb{Z}_q^*$ and want to agree on a key. They each broadcast their public keys (elements of G_1):

$$[A : aP \longrightarrow B, C][B : bP \longrightarrow A, C][C : cP \longrightarrow A, B]$$

- They then calculate $[A : e(bP, cP)^a][B : e(aP, cP)^b][C : e(aP, bP)^c]$
- Due to bilinearity, they share the secret key

$$e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c = e(P, P)^{abc} \in G_2$$

Bilinear DH Assumption

- This gives a new hardness assumption, **The Bilinear Diffie-Hellman (BDH) Assumption**:
- Given $\{P, aP, bP, cP\}$, the computation of $e(P,P)^{abc}$ is hard.

Pairings & ID-Based Encryption

Boneh & Franklin 01:

- **Setup.** (TTP) Again, $e: G_1 \times G_1 \rightarrow G_2$ a bilinear map and P a generator of G_1 with order q. TTP generates private key $v \in_R \mathbb{Z}_q^*$ and public key $V = vP \in G_1$. Public functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow G_1$ and $\mathsf{H}_2 : G_2 \rightarrow \{0,1\}^*$.
- **Keygen.** $W = v\mathsf{H}_1(ID) \in G_1$. ID is the user's identify and W the private key. The public key is *actually* ID!
- **Encrypt.** Given TTP's public key V, to encrypt the message m to identity ID:

$$
\begin{aligned}
Enc(V, ID, m) &= \{c_1, c_2\} \\
c_1 &= kP \text{ where } k \in_R \mathbb{Z}_q^* \\
c_2 &= m \oplus \mathsf{H}_2(e(\mathsf{H}_1(ID), V)^k)
\end{aligned}
$$

- **Decrypt.** To decrypt $\{c_1, c_2\}$ using private key W:

$$
\begin{aligned}
Dec(c_1, c_2, W) &= c_2 \oplus \mathsf{H}_2(e(W, c_1)) = c_2 \oplus \mathsf{H}_2(e(v\mathsf{H}_1(ID), kP)) \\
&= c_2 \oplus \mathsf{H}_2(e(\mathsf{H}_1(ID), P)^{vk}) = c_2 \oplus \mathsf{H}_2(e(\mathsf{H}_1(ID), vP)^k) \\
&= c_2 \oplus m \oplus c_2 = m
\end{aligned}
$$

Scheme Comments

- TTP can read everything.
- Encryption can take place before ID has a private key.
- How are H_1 and H_2 realized?

Elliptic Curves

- Elliptic curve $E(K): y^2 = x^3 + ax + b$ when $Char(K) \neq 2,3$.
- Form a group from the points (solutions) on the curve over K. (plus an identity element O).
- Group law:
  - *Identity*. $\forall P \in E$, $P + O = O + P = P$. (O is the point-at-infinity.)
  - *Negatives/Inverses*. $\forall P = (x,y) \in E$, $-P = (x,-y)$ and $P + -P = O$.
  - *Point Addition/Doubling*. With $P = (x_1,y_1)$ $Q = (x_2,y_2)$ $R = (x_3,y_3)$, $P + Q = R$ is:

$$x_3 = \lambda^2 - x_1 - x_2$$

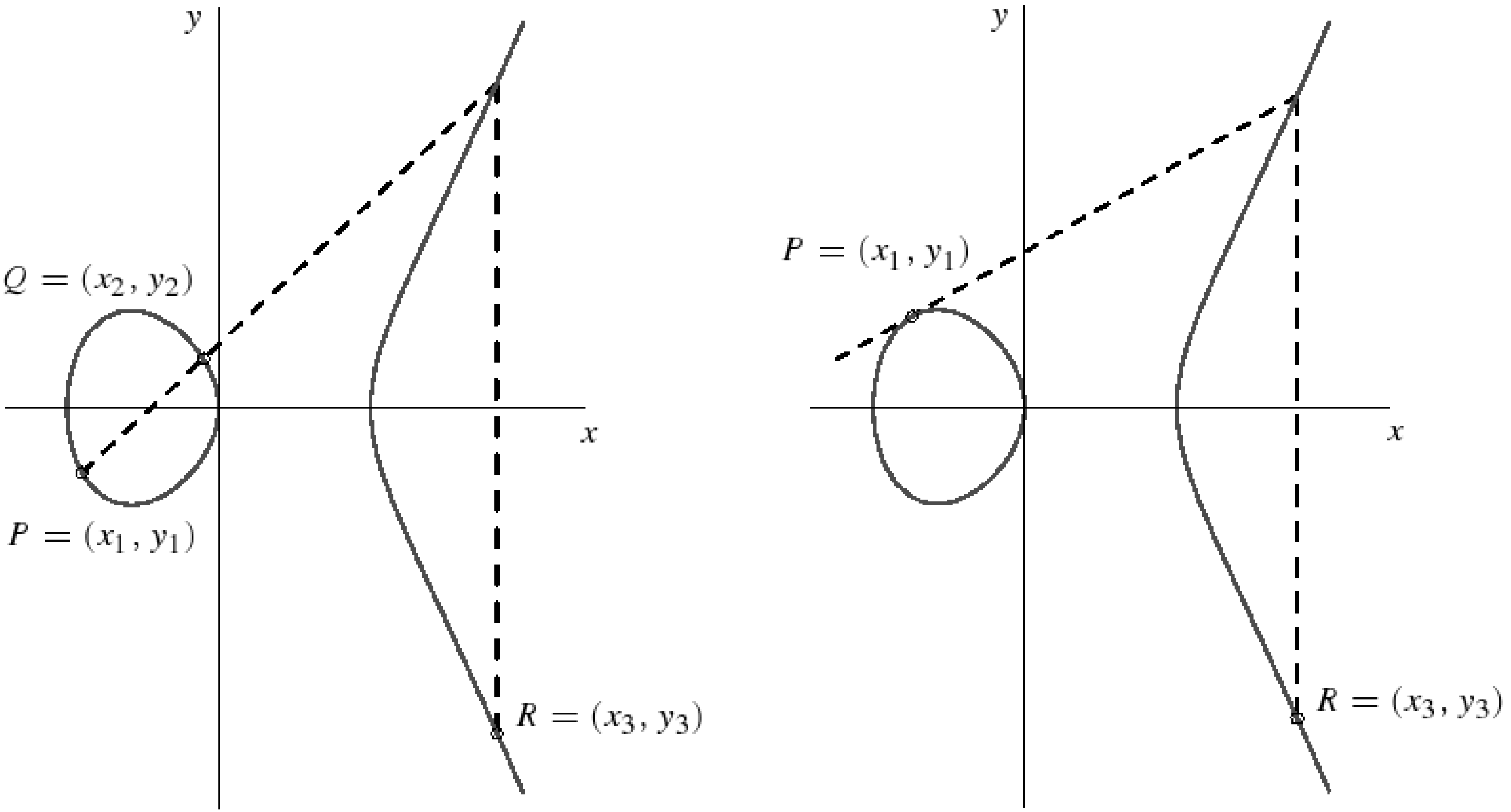
$$y_3 = \lambda(x_1 - x_3) - y_1$$, where λ is the slope:

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \text{ (point addition)} \\ \dfrac{3{x_1}^2 + a}{2y_1} & \text{if } P = Q \text{ (point doubling)} \end{cases}$$

- Scalar Multiplication: $kP = \sum_{i=0}^{\log_2 k} k_i 2^i P$
- (1986) N. Koblitz and V. Miller independently suggested elliptic curves for cryptographic use.

The Group Law Geometrically

(src: Hankerson, Menezes, Vanstone, Guide to Elliptic Curve Cryptography, Springer 04)



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.

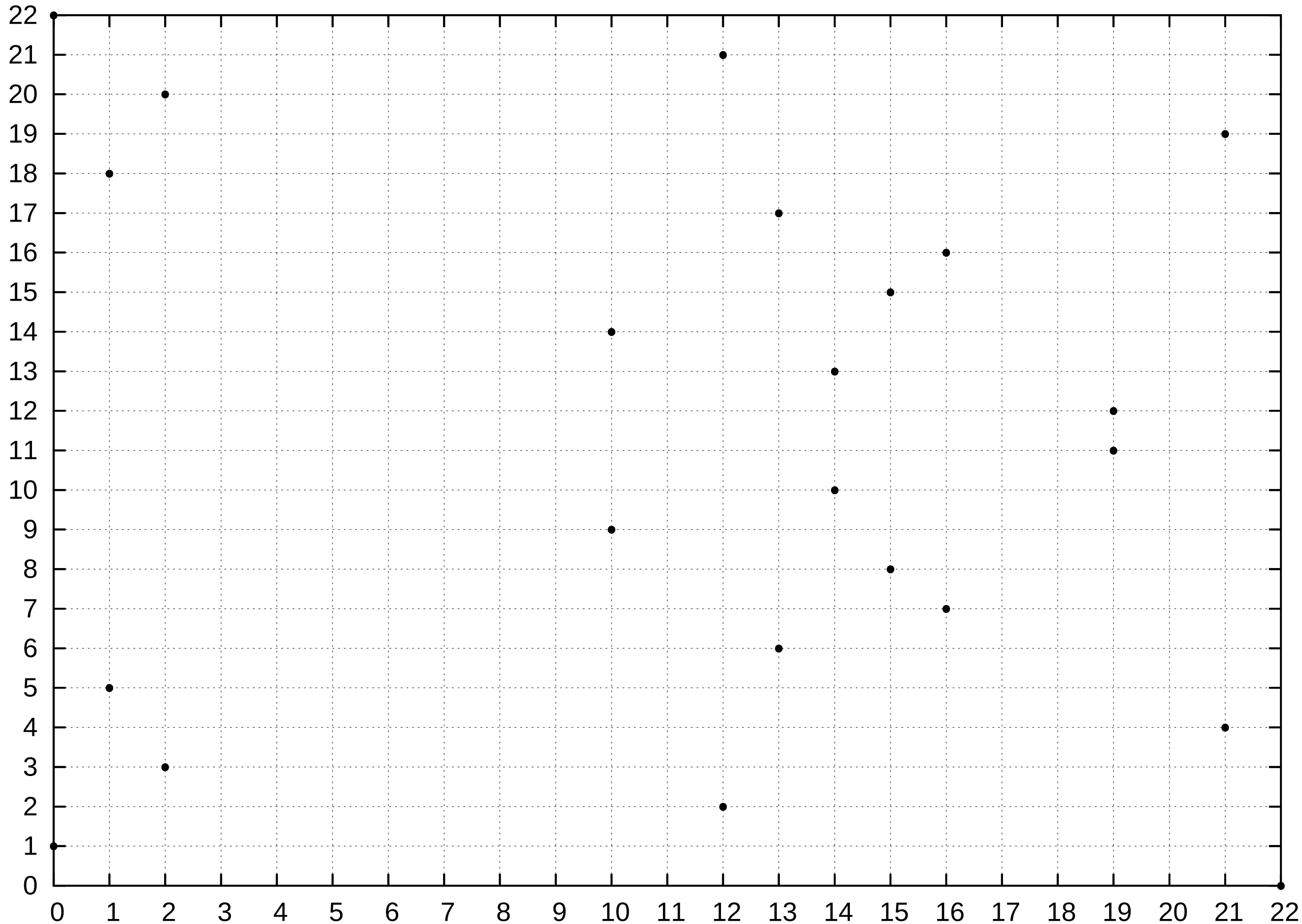
Elliptic Curves over a Finite Field

- What's the order of the curve? Well, logically $\#E \approx p \ldots$
- Hasse Bound: $p+1-2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p+1+2\sqrt{p}$, or...
- $\#E = p+1-t$ where t is "trace of the Frobenius." (small)
- if p divides t the curve is called **supersingular**.
- Example: $E(\mathbb{F}_{23}) : y^2 = x^3+1$ is cyclic; $P = (21,4)$ generates the entire group and $ord(P) = \#E = 24$. $t = 0$ so E is supersingular.

$1P$	(21,4)	$7P$	(14,10)	$13P$	(19,11)	$19P$	(13,6)
$2P$	(12,21)	$8P$	(0,1)	$14P$	(1,18)	$20P$	(2,20)
$3P$	(16,7)	$9P$	(10,14)	$15P$	(10,9)	$21P$	(16,16)
$4P$	(2,3)	$10P$	(1,5)	$16P$	(0,22)	$22P$	(12,2)
$5P$	(13,17)	$11P$	(19,12)	$17P$	(14,13)	$23P$	(21,19)
$6P$	(15,15)	$12P$	(22,0)	$18P$	(15,8)	$24P$	(0,0) = O

Elliptic Curve over a Finite Field (Fig.)

$E(\mathbb{F}_{23}) : y^2 = x^3 + 1$. $\#E = p + 1 - t = 23 + 1 - 0 = 24$ (23 points $+\mathcal{O}$)



Elliptic Curves & Discrete Logs

- *Pollard's Rho Algorithm*: G of order q, solves the general DLP in $\approx \sqrt{q}$ steps (exponential).
- *Index Calculus* (IC) for $\mathbb{Z}_p^*$: solves DLP in $\approx e^{(2+o(1))\sqrt{\log p \log\log p}}$ (subexponential).
- An IC analogue for solving ECDLP would try to "lift" points to the rationals $\mathbb{Q}$; the size of lifted points is not practical.
- Hence, the best algorithm for solving ECDLP is exponential—this is why we like ECC!

The Tate Pairing

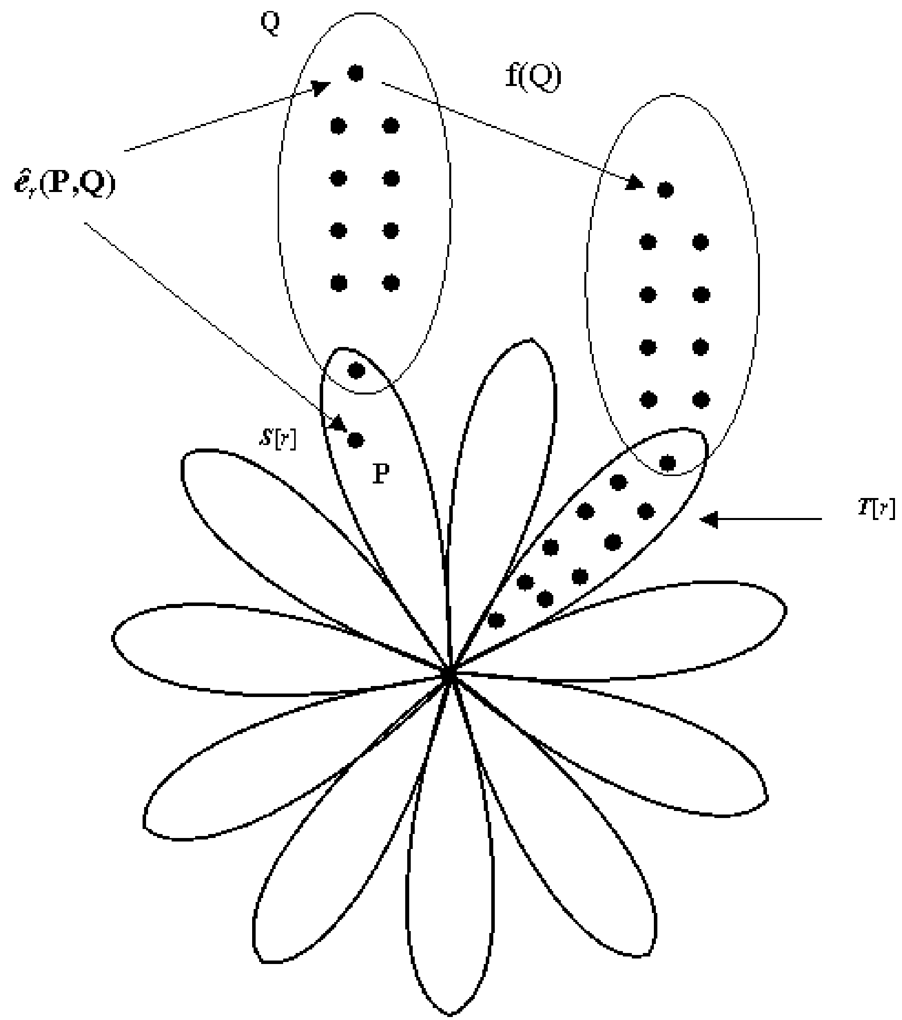
- For curves, the smallest positive integer k such that $p^k \equiv 1 \pmod{q}$ is the **embedding degree**. (Intuitively k is the multiplicative order of p modulo q.)
- For pairings, we want k small(ish). For random curves, k is probably big; for supersingular curves, $k \leq 6$. (See example curve $E(\mathbb{F}_{23})$.)
- The **Tate Pairing**

$$e : E(\mathbb{F}_p)[q] \times E(\mathbb{F}_{p^k})[q] \longrightarrow \mathbb{F}^*_{p^k}$$

  satisfies the following properties:
  - **Non-degeneracy.** $\forall P \in E(\mathbb{F}_p)[q] \setminus O \exists Q \in E(\mathbb{F}_{p^k})[q] | e(P,Q) \neq 1$.
  - **Bilinearity.** $\forall P \in E(\mathbb{F}_p)[q], Q \in E(\mathbb{F}_{p^k})[q], a \in \mathbb{Z}^*_q$, $e(aP,Q) = e(P,aQ) = e(P,Q)^a$.
- for k small, **this means**:
  - DLP methods can be used to solve ECDLP. (See Thm. 1.)
  - ID-Based Crypto with pairings is efficient.

Visualizing Pairings

(src: M. Scott, The Tate Pairing)



Miller's Algorithm

- Written by V. Miller in 1986, never formally published.
- Foundation of all modern pairing computation.
- Like scalar multiplication + some extras. Not particularly fast.

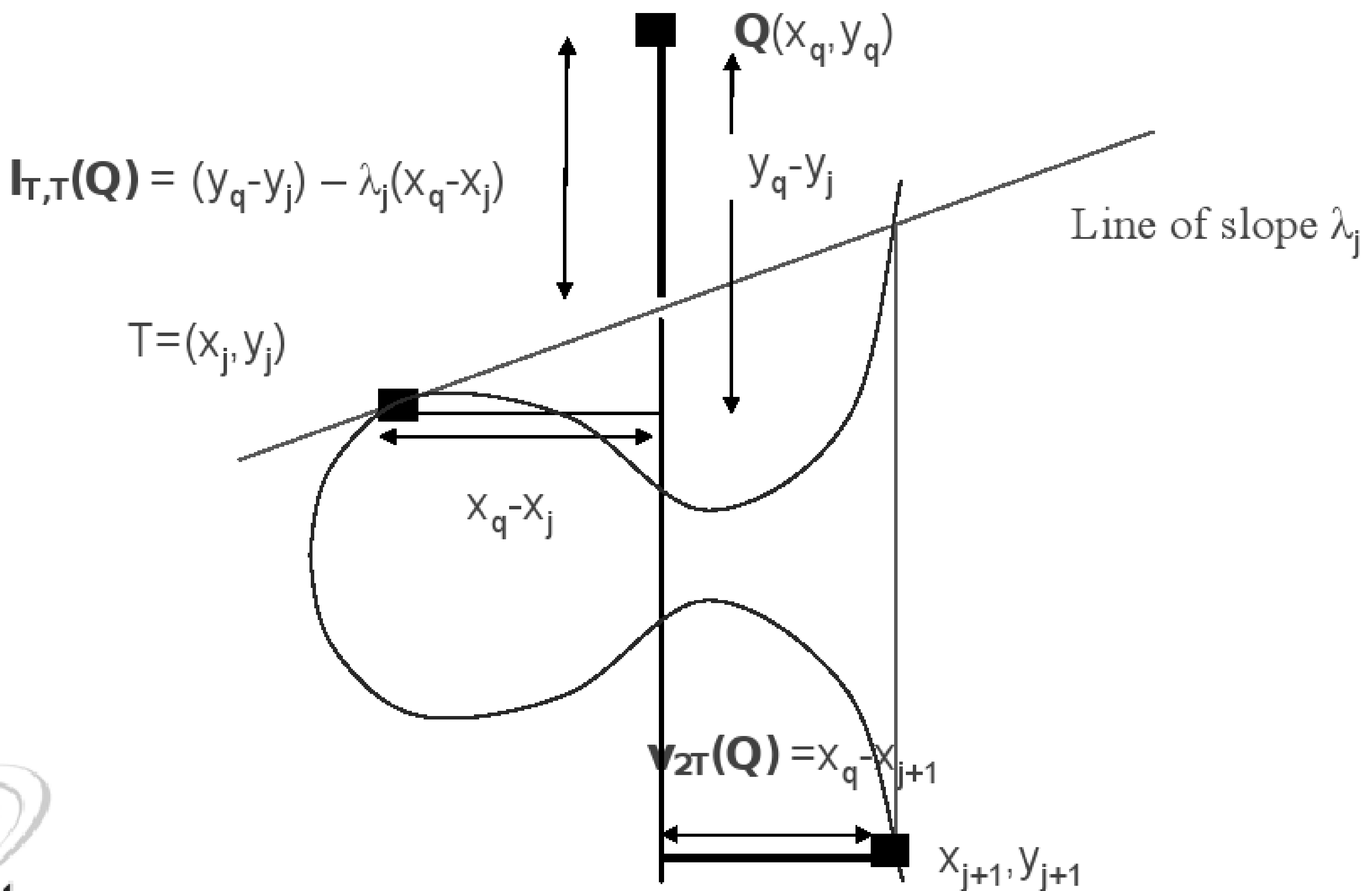
Input: group order q, points $P \in E(\mathbb{F}_p)$, $Q \in E(\mathbb{F}_{p^k})$
Output: Tate pairing evaluation, $e(P,Q) \in \mathbb{F}^*_{p^k}$

$f \leftarrow 1,\ T \leftarrow P$ /* $f \in \mathbb{F}_{p^k}$, $T \in E(\mathbb{F}_p)$ */

for $i \leftarrow \log_2 q - 1$ **to** 0 **do**
  $f \leftarrow f^2 \cdot l_{T,T}(Q)/v_{2T}(Q)$, $T \leftarrow 2T$
  if $q_i = 1$ **then** $f \leftarrow f \cdot l_{T,P}(Q)/v_{T+P}(Q)$, $T \leftarrow T + P$
end
return f

Visualizing Miller’s Algorithm

(src: M. Scott, Efficient Implementation of Cryptographic pairings)



Conclusion

- ID-Based Crypto is a great alternative for some environments. (Particularly, resource-constrained devices; wireless sensor networks?)
- ID-Based Crypto with pairings is compact and fun!
- Lack of supporting standards.
- References/Resources:

  - Canetti, Rivest, Special Topics in Cryptography, Lec. 25: Pairing-Based Cryptography, 04.
  - Mao, Modern Cryptography: Theory and Practice, 2003.
  - Hankerson, Menezes, Vanstone, Guide to Elliptic Curve Cryptography, Springer 04.
  - http://www.computing.dcu.ie/ mike/tate.html The Tate Pairing.
  - M. Scott, Presentation: Efficient Implementation of Cryptographic pairings.
  - Pairing-Based Crypto Lounge.
    http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html