Elliptic Curve Cryptography Pairing-based Cryptography: Applications and Optimizations

Kristin Lauter
Cryptography Group, Microsoft Research
June 16, 2009

Public Key Cryptography

- 1. Key exchange: two parties agree on a common secret using only publicly exchanged information
- 2. Signature schemes: allows parties to authenticate themselves
- Examples of public key cryptosystems:

  RSA, Diffie-Hellman, ECDH, DSA, ECDSA

Applications:

- Secure browser sessions (https: SSL/TLS)
- Signed, encrypted email (S/MIME)
- Virtual private networking (IPSec)
- Authentication (X.509 certificates)

Diffie-Hellman Key Exchange

Given a cyclic group G generated by g

Alice picks random a

Bob picks random b

Alice sends g^a →

← Bob sends g^b

Secret :

$$g^{ab} = (g^b)^a = (g^a)^b$$

Problem:

- Public key operations are computationally expensive compared to symmetric key (block ciphers, stream ciphers, DES, AES)
- Public keys can be long: currently in use 1024-bit RSA up to 16,000-bit keys
- Issues of power, bandwidth, and time

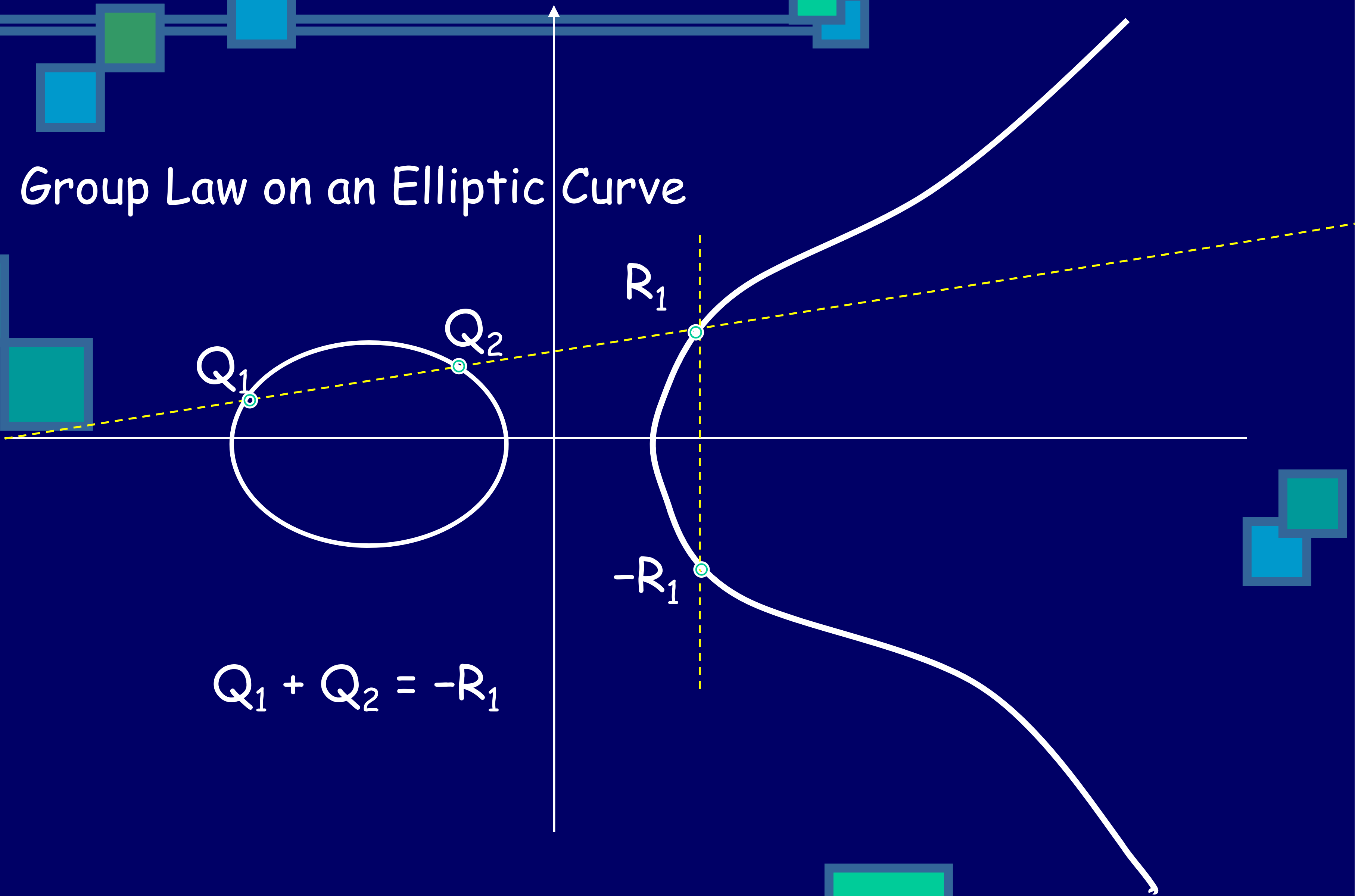
Elliptic Curve Cryptography

- Elliptic Curve Cryptography (ECC) is an alternative to RSA and Diffie-Hellman, primarily signatures and key exchange
- Proposed in 1985 (vs. 1975 for RSA)
- Security is based on a hard mathematical problem (different than factoring)

Group of points on an elliptic curve

- Traditional group: integers modulo prime with modular multiplication
- Minimum size of prime: 1024 or 2048 bits
- Alternative: group of points (x, y) on an elliptic curve, $y^2 = x^3 + a x + b$, modulo a prime of minimum size: 160 or 256-bits

Group Law on an Elliptic Curve



$$Q_1 + Q_2 = -R_1$$

Advantages over RSA/DH

- Shorter key lengths (for equivalent security levels against known attacks)

→

1) Fewer bits to store and send
2) Less computational power
3) Faster

Key length equivalences

symmetric	ECC	RSA/DH
80	163	1024
128	283	3072
192	409	7680
256	571	15360

(equal difficulty against currently known attacks)

Sample timing comparison

On Intel Pentium IV 1700Mhz :

Key length	Ratio RSA:ECC
RSA1024/ECC163	7:1
RSA3072/ECC283	60:1

Ratios more dramatic for special curves: 28 and 242

Implementations

- Marketed for mobile commerce by Certicom. (2003 NSA-Certicom deal)
- Implemented by Motorola, Sony, Lucent, RIM, Qualcomm, Verisign, OpenWave (Sun donated it to OpenSource)
- MSR-Crypto implemented ECC for MME (Microsoft Mobile Explorer) in June 2000, shipped in Vista 2007

U.S. Standards governing ECC

Draft ietf standards for ECC for

1) TLS, successor to SSL (secure browser)
2) S/MIME, CMS (secure email)
3) IPSec, X509 certificates, …

- FIPS, Digital Signature Standard (NIST)
- ANSI X9.62, X9.63 (Financial Services)
- IEEE P1363 (MS participating member)

NIST Curves

(National Institute of Standards and Technology)

- Standard curves for P-256, B-256, K-256
  - P- prime fields
  - B- binary fields
  - K- Koblitz curves (defined over F_2)
- Prime fields use special primes:

Generalized Mersenne Primes with very fast modular reduction

Binary exponentiation, NAF

To compute 7P = (111)P:

- 4P+2P+P (2 doubles, 2 add)
- 2(2P+P) + P (2 doubles, 2 add)
- different order
- left-to-right vs right-to-left

NAF=non-adjacent form

- Sparser expansion using subtractions:
- 7P = (100-1)P

Affine vs. Projective coordinates

- (weighted) Projective coordinates allow group law computation on E without field inversions, at the cost of more multiplies
- Roughly 16 field multiplies to compute 2P
- Better than affine if field inversions are very expensive
- e.g. for NIST prime curves, some estimate 1 inversion ~ 80 multiplies.
- MS implementation general curve, 1 I ~ 5M

Optimizations: 2P+Q

$P = (x_1, y_1)$ and $Q = (x_2, y_2)$

- x_1 not $= x_2$

$P + Q = (x_3, y_3)$

- $s = (y_2 - y_1)/(x_2 - x_1)$
- $x_3 = s^2 - x_2 - x_1$
- $y_3 = (x_1 - x_3)s - y_1$

2P+Q

Now add (P + Q) to P

Add (x_1, y_1) to (x_3, y_3)

Assume x_3 not = x_1.

- $2P+Q = (x_4, y_4)$
- $r = (y_3 - y_1)/(x_3 - x_1)$
- $x_4 = r^2 - x_1 - x_3$
- $y_4 = (x_1 - x_4)r - y_1$.

Omit y_3

- [Eisentraeger-L-Montgomery RSA03]
- We can omit the y3 computation, because it is used only in the computation of r
- $r = -s - 2y_1/(x_3 - x_1)$.
- Omitting the y_3 computation saves a field multiplication.
- Each formula requires a field division, so the overall saving is 1 field multiplication.

[Ciet-Joye-L-Montgomery 05]

- Trick extends to projective coordinates
- Extends to 3P+Q, ternary exponentiation
- mixed binary/ternary
- Can be used for multi-exponentiation:

to compute $k_1P_1 + k_2P_2$

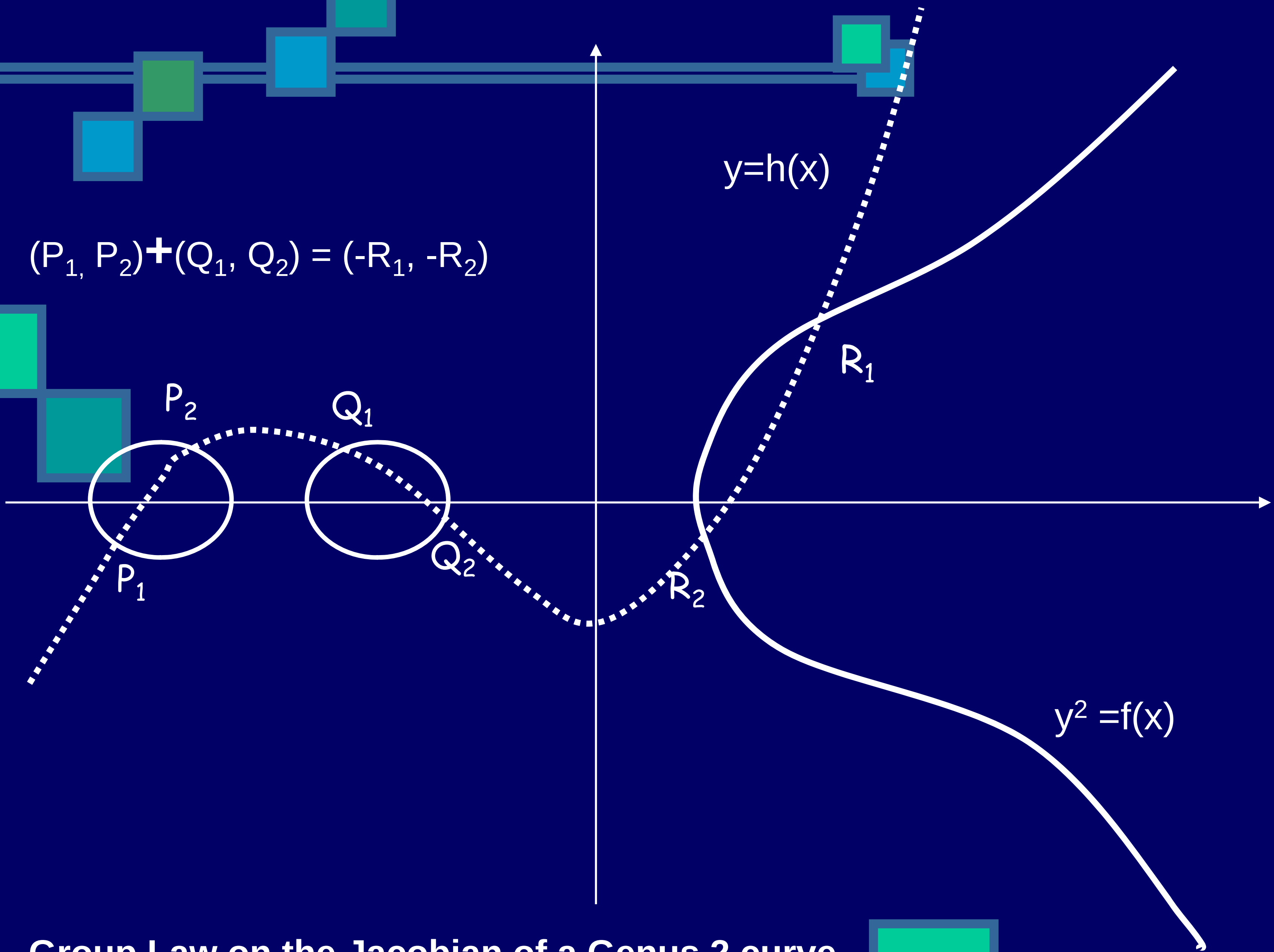
Table 1. Costs of simple operations on E

Doubling 2P	2 squarings, 1 multiplication, 1 division
Add P ± Q	1 squaring, 1 multiplication, 1 division
Double-add 2P ± Q	2 squarings, 1 multiplication, 2 divisions
Tripling 3P	3 squarings, 1 multiplication, 2 divisions
Triple-add 3P ± Q	3 squarings, 1 multiplication, 3 divisions

Another group for DLP: Jacobians of hyperelliptic curves

- Genus 2 curves given by the equation
- C: $y^2 = f(x)$, degree f = 5 or 6
- Group of points on the Jacobian J(C)
- Represented by pairs of points on C
- Efficient group law: Cantor’s algorithm

$$(P_1, P_2) + (Q_1, Q_2) = (-R_1, -R_2)$$



Group Law on the Jacobian of a Genus 2 curve

Pairings in Cryptography

- MOV attack on ECDLP

Menezes-Okamoto-Vanstone

- In 2001, Boneh-Franklin introduced IBE

Identity-Based Encryption

- Joux, Tri-partite Diffie-Hellman

Many other applications…

- ABE (attribute-based encryption)
- PEKS (Public Key Encryption with Keyword Search)
- Predicate Encryption …

BLS Short signatures: Boneh, Lynn, Shacham

Given a bilinear pairing (map):

$$e: G_1 \times G_1 \rightarrow G_2,$$

With a secret, x, a group element, P, in G_1, and a hash function h

1. Create a public key pair (P, Q=xP)
2. Sign messages M → (M, S(M)), S(M) = x h(M)
3. Verification is: e(Q,h(M)) = e(P,S(M)) ?
   bilinearity → **e(xP,h(M)) = e(P,xh(M))**

Implemented using Weil or Tate pairing, when G_1 is an elliptic curve and G_2 is the multiplicative group of a finite field

Pairings

- Weil pairing on elliptic curves
- Tate pairing on elliptic curves
- Squared Weil and Tate pairings
- Ate pairing
- Eta pairing and generalized forms

All these for Jacobians hyperelliptic curves:
[Duursma-Lee 03], [ELM 04], [Lee et al]

Computing functions for Miller's loop

- To compute $e_m(P,Q)$: P,Q in E[m]
- Find a function f_c on E with a c-fold zero at P, a simple pole at cP , a pole of order c – 1 at O, and no other zeroes or poles.
- Compute f_m recursively
- $(f_m) = mP - mO$
- $e_m(P,Q) = f_m(Q_1)/f_m(Q_2)$

Recursive step

- Compute f_{b+c} , f_{b-c} from (f_b, bP), (f_c, cP)
  - $g_{b,c}$ = line through bP and cP
  - g_{b+c} = vertical line through (b+c)P

- $f_{b+c} = f_b \cdot f_c \cdot g_{b,c}/g_{b+c}$
- $f_{b-c} = f_b \cdot g_b / (f_c \cdot g_{-b,c})$

- Denote $h_b = f_b(Q_1)/f_b(Q_2)$

Parabola trick for pairings

[Eisentraeger-L-Montgomery RSA03]

$(h_{2b+c}, (2b + c)P)$ given (h_b, bP), (h_c, cP)

Compute $(h_{2b+c}, (2b + c)P)$ directly, only the x-coordinate of bP + cP

$f_{2b+c} = f_{b+c} \cdot f_b \cdot g_{b+c,b} / g_{2b+c}$

- $= f_b \cdot f_c \cdot g_{b,c} \cdot f_b \cdot g_{b+c,b} / (g_{2b+c}\, g_{b+c})$
- $= f_b \cdot f_c \cdot f_b / (g_{2b+c}) \cdot g_{b+c,b} \cdot g_{b,c} / (g_{b+c})$

Parabola

- replace $g_{b+c,b} \cdot g_{b,c} /(g_{b+c})$ by a parabola through the points

$$bP\ ,\ bP\ ,\ cP\ ,\ -(2b+c)P$$

- $(x-x_1)(x+x_1+x_3+rs) - (r+s)(y-y_1)$.
- Note: do not compute y_3
- Evaluate the formula for f_{2b+c} at Q_1 and Q_2 to get a formula for h_{2b+c}.
- Saving 8-12% overall.

Squared Pairings

- Eisentraeger-L-Montgomery
- Can compute the pairing without using a random R

$$Q\text{-}O \sim (Q+R) - R$$

Get some denominator cancellation

$\text{Tate}_m(P,Q)^2 = (f_m(Q)/f_m(-Q))^{q-1/m}$

20% performance improvement

Works for hyperelliptic curves, too.

Digital Signatures for Network Coding

Kristin Lauter, Microsoft Research
Joint work with: Denis Charles, Kamal Jain (Microsoft)

NIMS Workshop on Mathematical Cryptology
June 16, 2009

The Network Coding Model

Let $G = (V, E)$ be a directed graph.

The vertices $v \in V$ may represent, for example, peers in a peer-to-peer content distribution or content storage system.

A source $s \in V$ wishes to transmit content to a set $T \subseteq V(G)$ of receivers in the system using network coding.

The file to be transmitted is broken up into blocks of equal length and represented as vectors in an $\mathbb{F}_p$-vector space of dimension d.

$$\mathbf{w}_1, \cdots, \mathbf{w}_k \in \mathbb{F}_p^d$$

Tags for network coding

Network coding was introduced by: [Alswede, Cai, Li, Yeung 2000], [Chou, Jain, Wu 2003], [Gkantsidis, Rodriguez 2005], ...

The source transmits the augmented vectors to its neighbors.

$$\mathbf{v}_i = \langle \underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0, w_{i1}, \ldots, w_{id} \rangle \in \mathbb{F}_p^{k+d} \text{ for } 1 \leq i \leq k$$

the ith entry of v_i is a 1, and $\mathbf{w}_i = (w_{i1}, \ldots, w_{id})$

Recombination

Each edge e, computes

$$\mathbf{y}(e) = \sum_{f:\mathrm{out}(f)=\mathrm{in}(e)} m_e(f)\mathbf{y}(f),$$

where $m_e(f) \in \mathbb{F}_p$.

Another peer receives $\mathbf{y}(e)$ and then continues to recombine it with other received inputs.

Recovery of original content

If a receiver $t \in T$ gets

$$\mathbf{u}_1 = \langle g_{11}, \cdots, g_{1k}, u_{11}, \cdots, u_{1d} \rangle$$
$$\mathbf{u}_2 = \langle g_{21}, \cdots, g_{2k}, u_{21}, \cdots, u_{2d} \rangle$$
$$\vdots$$
$$\mathbf{u}_k = \langle g_{k1}, \cdots, g_{kk}, u_{k1}, \cdots, u_{kd} \rangle$$

then t can find $\mathbf{w}_1, \cdots, \mathbf{w}_k$ by solving

$$\begin{pmatrix} u_{11} & \cdots & u_{1d} \\ u_{21} & \cdots & u_{2d} \\ & \vdots & \\ u_{k1} & \cdots & u_{kd} \end{pmatrix} = \begin{pmatrix} g_{11} & \cdots & g_{1k} \\ g_{21} & \cdots & g_{2k} \\ & \vdots & \\ g_{k1} & \cdots & g_{kk} \end{pmatrix} \begin{pmatrix} w_{11} & \cdots & w_{1d} \\ & \vdots & \\ w_{k1} & \cdots & w_{kd} \end{pmatrix}.$$

Pollution attacks

Network coding for peer-to-peer content distribution improves throughput since there are no "bottlenecks".

No peers are left waiting for the last piece of the file, since almost any subsequent linear combination of the pieces will contain new information which can be used to reconstruct the file, until the peer has received enough pieces of information.

The problem with network coding is that it is very suceptible to pollution attacks, since garbage packets are quickly recombined with other "clean" packets and redistributed to pollute the whole network.

The Signature Scheme

Let $E/\mathbb{F}_q$ be an elliptic curve and let

$$R_1, \cdots, R_k, P_1, \cdots, P_d \in E(\mathbb{F}_q)[p]$$

p-torsion points on E: $pR_i = pP_j = 0$ for $1 \leq i \leq k, 1 \leq j \leq d$.
Define a function $h_{R_1,\cdots,R_k,P_1,\cdots,P_d} : \mathbb{F}_p^{k+d} \to E(\mathbb{F}_q)$ by

$$h_{R_1,\cdots,R_k,P_1,\cdots,P_d}(u_1, \cdots, u_k, v_1, \cdots, v_d) = \sum_{1\leq i\leq k} u_i R_i + \sum_{1\leq j\leq d} v_j P_j.$$

Signing vectors

The source s selects $s_1, \cdots, s_k, r_1, \cdots, r_d$ and signs the vector

$$\mathbf{v}_i = \langle \underbrace{0, \cdots, 0}_{i-1}, 1, w_{i1}, \cdots, w_{ik} \rangle \in \mathbb{F}_p^{k+d} \text{ for } 1 \leq i \leq k$$

by computing

$$\sigma_i = h_{s_1 R_1, \cdots, s_k R_k, r_1 P_1, \cdots, r_d P_d}(\mathbf{v}_i).$$

Source also publishes $Q, s_1 Q, \cdots, s_k Q, r_1 Q, \cdots, r_d Q$ where Q is another p-torsion point such that $\mathbf{e}(R_i, Q) \neq 1$ and $\mathbf{e}(P_i, Q) \neq 1$.

Recombining signed vectors

Now σ_i is transmitted together with $\mathbf{v}_i$ to the neighbors of the source s. Each edge e computes

$$\mathbf{y}(e) = \sum_{f:\mathrm{out}(f)=\mathrm{in}(e)} m_e(f)\mathbf{y}(f).$$

and

$$\sigma(e) = \sum_{f:\mathrm{out}(f)=\mathrm{in}(e)} m_e(f)\sigma(f).$$

Verification

Suppose $\mathbf{y}(e) = \langle u_1, \cdots, u_k, v_1, \cdots, v_d \rangle$ we check whether

$$\prod_{1 \le j \le k} \mathbf{e}(u_j P_j, s_j Q) \prod_{1 \le i \le d} \mathbf{e}(v_i P_i, r_i Q) = \mathbf{e}(\sigma(e), Q).$$

Hardness assumptions

Fact: [CJL'05] Finding a collision of the hash function h is polynomial-time equivalent to computing the discrete log on the elliptic curve E.

Fact: Forging signatures is as hard as the computational Diffie-Hellman problem on the curve E.

Remarks

If we take the prime $p \approx 256$-bits, this is equivalent to 2048 bits of RSA security. We can setup the system with $q \approx p^2$.

- ⋆ Our scheme establishes authentication in addition to security.
- ⋆ Communication overhead per vector is two elements of $\mathbb{F}_p$ (the x and y coordinates of a point) $=$ 512 bits. We can reduce this overhead to 257 bits at the cost of increasing computational cost.
- ⋆ Computation of signature of vector at an edge e is $O(\mathrm{indeg}(\mathrm{in}(e))$ operations in $\mathbb{F}_p$.
- ⋆ Verification requires $O((d+k)\log^{2+\epsilon} q)$ bit operations for any $\epsilon > 0$.
- ⋆ Scheme requires p of size 256-bits.

Example

$p = 263300183685717422065746325660655084022315089991 53.$

$\ell = p^2 + 1875150302622039835263003517434470200231290230217730^2$

$= 35168819272908168996348622156834481670445567551962199157 26$

$5479286004610264134079797473542444269610703 09.$

The complex multiplication method tells us that the elliptic curve

$$E : y^2 = x^3 + x \text{ (in affine form)}$$

is a suitable elliptic curve. MAGMA tells us that $\#E(\mathbb{F}_\ell)$ is

$35168819272908168996348622156834481670445567551962198630665111$

$9766132641428476163374399639430720 04,$

which is indeed $\equiv 0 \mod p$.

This computation took 0.063 seconds on an AMD Opteron 252 (2.6Ghz) processor.

Example (continued)

The number of points on $E(\mathbb{F}_{\ell^2})$ according to MAGMA is

1236845849050477072586861412005782314655826646818745936122594

0144846014265383739300784290963417699135578021643493118755085

0388577638414226886949389446808131945333677281203696574462646

and this is $\equiv 0 \mod p^2$, which is a necessary condition for $E[p]$ being a subgroup of $E(\mathbb{F}_{\ell^2})$.

We show that $E[p]$ is indeed contained in $E(\mathbb{F}_{\ell^2})$ by finding two points that generate the p-torsion subgroup. We find two p-torsion points, P and Q, that generate the whole p-torsion of $E(\mathbb{F}_{\ell^2})$

Example (continued)

$$P = (27670104998350953223410633845208244029271176277346373253$$
$$81486020583308437632397697221548 62, 73689561907486287044199 32$$
$$123419527006199990201373312978349862216019407508187132975485$$
$$Q = (170343693342782875614389009934880452275069084044323551860$$
$$4957564303078396992524604785250333u + 1571288746986618549950$$
$$152507760097756731298637781743699698629138614858935315679990$$
$$293262979414624776596432402939618431893907517428095829765520$$
$$72565240814005665686795414190u + 2827229136528454163001184937$$
$$521916237377189328124466481421733687054166538367154312288563$$

Example (continued)

Here u is a variable that gives the isomorphism $\mathbb{F}_{\ell^2} \cong \mathbb{F}_\ell[u]/(f(u))$ for a quadratic irreducible $f \in \mathbb{F}_\ell[u]$. The Weil pairing of P and Q is

$$\begin{aligned}\mathbf{e}_p(P, Q) = {} & 188036180299835372546533903820354629932054094777699 \\ & 3766041577935958159317265607540618580827 5672u + \\ & 31284655683961117025378938265048897550540714 \\ & 78912095275807108199402549356171889616725860797979581 9\end{aligned}$$