Slides for AMS Columbus talk,
to be given 2001.09.22.

Paper: "Faster square roots in annoying finite fields" (without the discussion of cryptographic applications).

Elliptic curve cryptography: the case of NIST P-224

D. J. Bernstein
University of Illinois at Chicago
NSF CCR–9983950

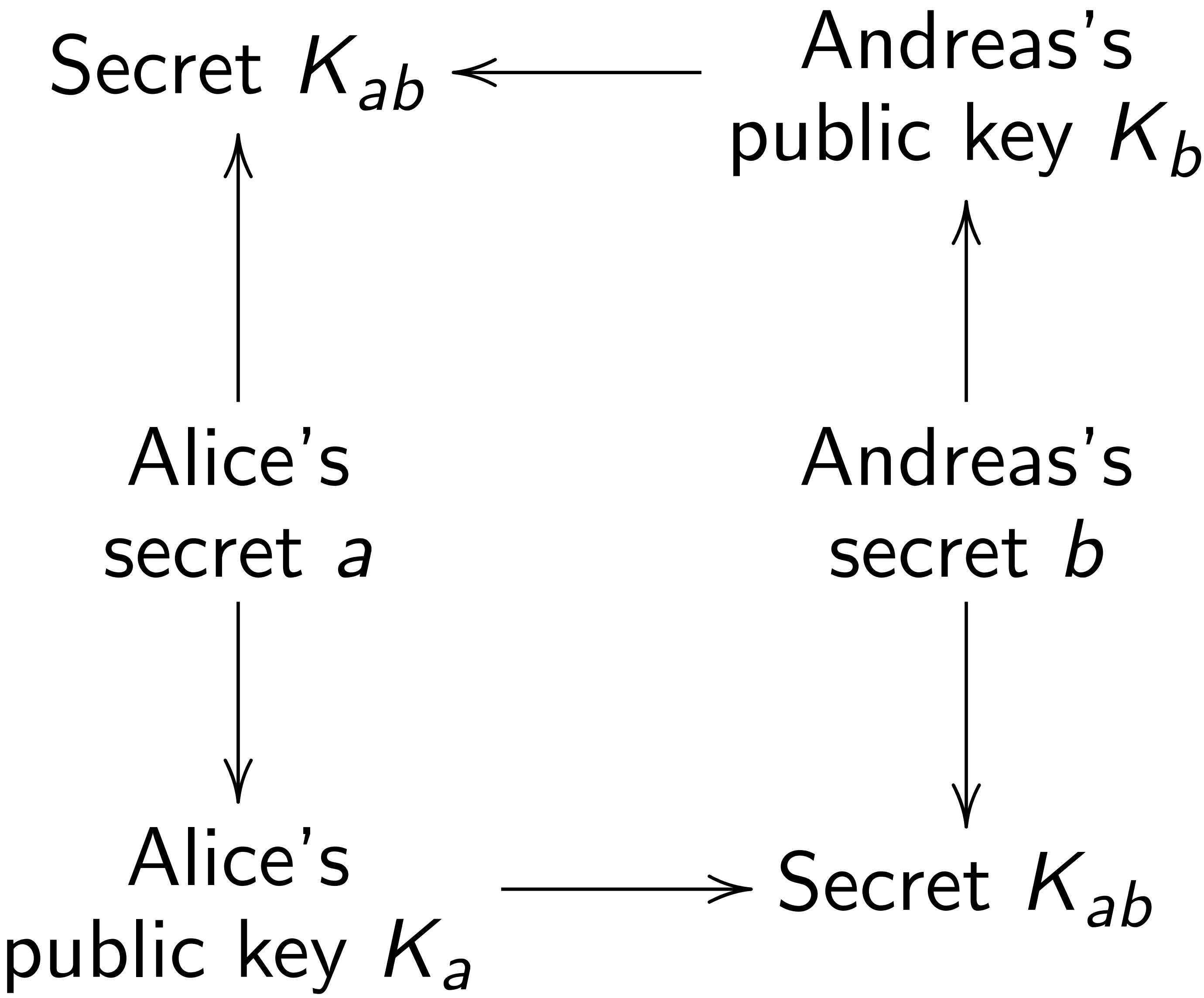
NIST P-224 is the elliptic curve $y^2 = x^3 - 3x + c_6$ over $\mathbf{Z}/p$.

Here $c_6 =$ 18958286285566608 00040866854449392 64155046809686793 21075787234672564

and $p = 2^{224} - 2^{96} + 1$.

Multiply $(10(2^{224}-1)/(2^8-1), \ldots)$ by n on the curve to get $(K_n, \ldots)$, for $n \in (\mathbf{Z}/\#\text{curve}(\mathbf{Z}/p))^*$.

The Diffie-Hellman system



Expand shared secret K_{ab} into long string of secrets: e.g., $\mathrm{SHA}(K_{ab}, 1), \mathrm{SHA}(K_{ab}, 2), \ldots$.

Use this string to authenticate and encrypt communications between Alice and Andreas.

`nistp224` is a new program to compute K_{ab} given a, K_b.

Alice puts 28 random bytes into `A`, 28 newlines into `K1`.

```
cat A K1 | nistp224 > KA
cat A KB | nistp224 > KAB
```

Also a C-language library.

Cycle counts to multiply by a given x or given x, y:

x	x, y	
752549	651953	Athlon
930174	813405	PPro/PII/PIII
1095312	951712	P4
1356615	1188130	P1/PMMX

First step: Given x,
compute a square root y of
$u = x^3 - 3x + c_6$ in $\mathbf{Z}/p$.

Cipolla's algorithm (1903):
Try random r's until finding
that $\Delta = r^2 - 4u$ is not a square.
Compute $y = ((t + r)/2)^{(p+1)/2}$
in $(\mathbf{Z}/p)[t]/(t^2 - \Delta)$.

Can compute $(p+1)/2$ power using 222 squarings, a few more mults.

Each squaring in $(\mathbf{Z}/p)[t]/(t^2-\Delta)$ involves 4 mults in $\mathbf{Z}/p$: 2 squarings, 1 mult by Δ, 1 more. Choose r to make Δ small.

> 900 mults in $\mathbf{Z}/p$ to find y.

Tonelli's algorithm (1891):

Precompute g of order 2^{96}.
For example: $g = 11^{(p-1)/2^{96}}$.

Compute $v = u^{(p-2^{96}-1)/2^{97}}$.
Then $uv^2 = u^{(p-1)/2^{96}}$ is
a power g^e with $e \in 2\mathbf{Z}$.
Compute e, bit by bit.
Compute $y = uvg^{-e/2}$.

Precompute $g^{-2^{6i}j}$
for $0 \le i \le 15$, $0 \le j \le 63$.
1024 precomputed values.

$g^{-e/2} = g^{-d_0} g^{-2^6 d_1} \ldots g^{-2^{90} d_{15}}$
if $e/2 = d_0 + 2^6 d_1 + \cdots + 2^{90} d_{15}$.

(Yao 1976, Pippenger 1980)

Discrete logs, bit by bit

Say $e = e_0 + 2e_1 + 4e_2 + \cdots$.
Given g^e, and given $e \bmod 2^k$, determine e_k from
$g^{2^{95} e_k} = (g^e g^{-(e \bmod 2^k)})^{2^{95-k}}$.

Thousands of multiplications for 2^{94} power, 2^{93} power, etc.

Save whenever $e_k = 0$. (Shanks)

Discrete logs, 6 bits at a time

Say $e = e_0 + 2^6 e_1 + \cdots + 2^{90} e_{15}$.

Given g^e:

Compute $g^{2^6 e}, g^{2^{12} e}, \ldots, g^{2^{90} e}$.

$g^{2^{90} e_k} = g^{2^{90-6k} e} g^{-2^{90-6k} e_0}$
$g^{-2^{90-6(k-1)} e_1} \ldots g^{-2^{90-6} e_{k-1}}$.

Can sort or hash powers of $g^{2^{90}}$.

364 mults to compute $u \mapsto y$.

Asymptotics

Square roots in $\mathbf{F}_q$, after polynomial-time precomputation.

Cipolla: $(4 + o(1)) \lg q$ mults.

Tonelli: $(1 + o(1)) \lg q$ mults, if $\operatorname{ord}_2(q-1) \in o(\sqrt{\lg q})$.

New: $(1 + o(1)) \lg q$ mults, if $\operatorname{ord}_2(q-1) \in o(\sqrt{\lg q} \lg \lg q)$.

Also usable for Pohlig-Hellman.