Practical Aspects of Modern Cryptography

Josh Benaloh & Brian LaMacchia

Lecture 7: Multi-Party Protocols and Interactive Proofs

A *bit* more on certificates

- X.509 is not the only certificate standard – see also X9.55, X9.57, X9.59, Xcetera, Xcetera, Xcetera.
- Several “web of trust” designs exist – in particular, see SPKI/SDSI.

Attribute Certificates

Anyone who has the private key associated with the included public key has the right to …

And now for something completely different.

Multi-Party Protocols

Thusfar, the protocols we’ve explored have dealt primary with two-party scenarios.

Many scenarios concern fair agreement and computation with more players.

Fair Selection

Suppose that a group wants to make a fair choice between two or more options.

How can this be done in an unbiased manner?

Secret Sharing

Suppose that I have some data that I want to share amongst three people such that

- any two can uniquely determine the data
- but any one alone has *no information whatsoever* about the data.

Secret Sharing

Some simple cases: "AND"

I have a secret value z that I would like to share with Alice and Bob such that both Alice *and* Bob can together determine the secret at any time, but such that neither has any information individually.

Secret Sharing – AND

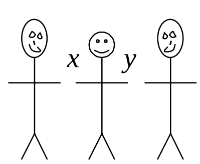
Let $z \in Z_n = \{0,1,\ldots,m-1\}$ be a secret value to be shared with Alice and Bob.

Randomly and uniformly select values x and y from Z_m subject to the constraint that

$$(x + y) \bmod m = z.$$

Secret Sharing – AND

The secret value is $z = (x + y) \bmod m$.



Secret Sharing – AND

This trick easily generalizes to more than two shareholders.

A secret S can be written as

$$S = (s_1 + s_2 + \ldots + s_n) \bmod m$$

for any randomly chosen integer values $s_1, s_2, \ldots, s_n$ in the range $0 \le s_i < m$.

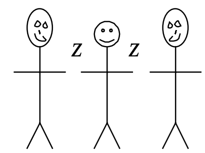
Secret Sharing

Some simple cases: “OR”

I have a secret value z that I would like to share with Alice and Bob such that either Alice *or* Bob can determine the secret at any time.

Secret Sharing – OR

The secret value is z.



Secret Sharing – OR

This case also generalizes easily to more than two shareholders.

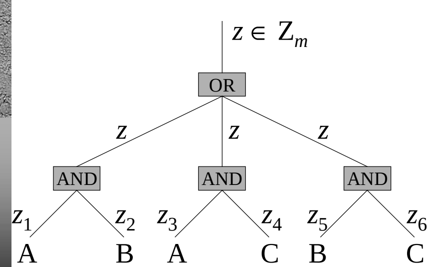
Secret Sharing

More complex *access structures* …

I want to share secret value z amongst Alice, Bob, and Carol such that any two of the three can reconstruct z.

$$S = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$$

Secret Sharing



Threshold Schemes

I want to distribute a secret datum amongst n trustees such that

- any k of the n trustees can uniquely determine the secret datum,
- but any set of fewer than k trustees has *no information whatsoever* about the secret datum.

Threshold Schemes

OR ≡ 1 out of n

AND ≡ n out of n

Shamir's Threshold Scheme

Any k points in a field *uniquely* determine a polynomial of degree at most k-1.

This not only works of the reals, rationals, and other infinite fields, but also over the finite field $Z_p = \{0,1,\ldots,p-1\}$ where p is a prime.

Shamir's Threshold Scheme

To distribute a secret value $s \in Z_p$ amongst a set of n Trustees $\{T_1,T_2,\ldots,T_n\}$ such that any k can determine the secret

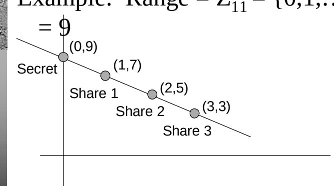
- pick random *coefficients* $a_1,a_2,\ldots,a_{k-1} \in Z_p$
- let $P(x) = a_{k-1}x^{k-1} + \ldots + a_2x^2 + a_1x + s$
- give $P(i)$ to trustee T_i.

The secret value is $s = P(0)$.

Shamir's Threshold Scheme

The threshold 2 case:

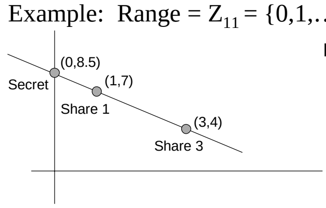
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

Example: Range = $Z_{11} = \{0,1,\ldots,10\}$



In Z_{11}, 8.5
$\equiv 17 \div 2$
$\equiv 6 \times 6$
$\equiv 36$
$\equiv 3$

Shamir's Threshold Scheme

Two methods are commonly used to interpolate a polynomial given a set of points.

- Lagrange interpolation
- Solving a system of linear equations

Lagrange Interpolation

- For each point (i,P(i)), construct a polynomial P_i with the correct value at i and a value of zero at the other given points.

$$P_i(x) = P(i) \times \prod_{(j \neq i)}(x-j) \div \prod_{(j \neq i)}(i-j)$$

- $P(x) = \sum_i P_i(x)$

Solving a Linear System

- Regard the polynomial coefficents as unknowns.
- Plug in each known point to get a *linear* equation in terms of the unknown coefficients.
- Once there are as many equations as unknowns, use linear algebra to solve the system of equations.

Verifiable Secret Sharing

Secret sharing is very useful when the "dealer" of a secret is honest, but what bad things can happen if the dealer is potentially dishonest?

Can measures be taken to eliminate or mitigate the damages?

Homomorphic Encryption

Recall that with RSA, there is a multiplicative *homomorphism*.

$$E(x)E(y) \cong E(xy)$$

Can we find an encryption function with an additive homomorphism?

An Additive Homomorphism

Can we find an encryption function for which the sum (or product) of two encrypted messages is the (an) encryption of the sum of the two original messages?

$$E(x) \circ E(y) \cong E(x+y)$$

An Additive Homomorphism

Recall the one-way function given by

$$f(x) = g^x \bmod m.$$

For this function,

$$f(x)f(y) \bmod m = g^x g^y \bmod m =$$
$$g^{x+y} \bmod m = f(x+y) \bmod m.$$

Verifiable Secret Sharing

- Select a polynomial with secret a_0 as
  $P(x) = a_{k-1}x^{k-1} + \ldots + a_2x^2 + a_1x + a_0$.
- Commit to the coefficients by publishing
  $g^{a_0}, g^{a_1}, g^{a_2}, \ldots, g^{a_{k-1}}$.
- Compute a commitment to $P(i)$ from public values as
  $g^{P(i)} = g^{a_0 i^0} g^{a_1 i^1} g^{a_2 i^2} \ldots g^{a_{k-1} i^{k-1}}$.

Verifiable Secret Sharing

An important detail

Randomness must be included to prevent small spaces of possible secrets and shares from being exhaustively searched.

Secret Sharing Homomorphisms

All of these secret sharing methods have an additional useful feature:

If two secrets are separately shared amongst the same set of people in the same way, then the sum of the individual shares constitute shares of the sum of the secrets.

Secret Sharing Homomorphisms

OR

Secret: a – Shares: $a, a, \ldots, a$
Secret: b – Shares: $b, b, \ldots, b$

Secret sum: $a+b$
Share sums: $a+b, a+b, \ldots, a+b$

Secret Sharing Homomorphisms

AND

Secret: a – Shares: $a_1, a_2, \ldots, a_n$
Secret: b – Shares: $b_1, b_2, \ldots, b_n$

Secret sum: $a+b$
Share sums: $a_1+b_1, a_2+b_2, \ldots, a_n+b_n$

Secret Sharing Homomorphisms

THRESHOLD

Secret: $P_1(0)$ – Shares: $P_1(1), P_1(2), \ldots, P_1(n)$
Secret: $P_2(0)$ – Shares: $P_2(1), P_2(2), \ldots, P_2(n)$

Secret sum: $P_1(0) + P_2(0)$
Share sums: $P_1(1) + P_2(1), P_1(2) + P_2(2), \ldots, P_1(n) + P_2(n)$

Verifiable Secret-Ballot Elections

- In an election, each voter can cast a vote by sharing the vote with a set of election officials at a pre-determined threshold.
- The officials can read an individual's vote only if a sufficiently large set conspire.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	S_{A1}	S_{A2}	S_{A3}
B	V_B	S_{B1}	S_{B2}	S_{B3}
C	V_C	S_{C1}	S_{C2}	S_{C3}
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

The *sum* of the *shares* of the votes constitute *shares* of the *sum* of the votes.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	S_{A1}	S_{A2}	S_{A3}
B	V_B	S_{B1}	S_{B2}	S_{B3}
C	V_C	S_{C1}	S_{C2}	S_{C3}
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

The shares of the votes can each be encrypted with an additively homomorphic encryption function.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	S_{A1}	S_{A2}	S_{A3}
B	V_B	S_{B1}	S_{B2}	S_{B3}
C	V_C	S_{C1}	S_{C2}	S_{C3}
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

To get encryptions of the sums, compute the products of the encryptions.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$ $E_1(\Sigma S_{i1})$	$\prod E_2(S_{i2})$ $E_2(\Sigma S_{i2})$	$\prod E_3(S_{i3})$ $E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Decrypt the products to determine the column sums.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
		$E_1(\Sigma S_{i1})$	$E_2(\Sigma S_{i2})$	$E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
		$E_1(\Sigma S_{i1})$	$E_2(\Sigma S_{i2})$	$E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Combine the shares to form the tally.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
		$E_1(\Sigma S_{i1})$	$E_2(\Sigma S_{i2})$	$E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
		$E_1(\Sigma S_{i1})$	$E_2(\Sigma S_{i2})$	$E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Verifiable Secret-Ballot Elections

Product of Encryptions ≡ Encryption of Sum
Sum of Shares ≡ Shares of Sum

The *product* of the *encryptions* of the *shares* of the votes constitute *encryptions* of the *shares* of the *sum* of the votes.

Verifiable Secret-Ballot Elections

Voter	Vote	Official 1	Official 2	Official 3
A	V_A	$E_1(S_{A1})$	$E_2(S_{A2})$	$E_3(S_{A3})$
B	V_B	$E_1(S_{B1})$	$E_2(S_{B2})$	$E_3(S_{B3})$
C	V_C	$E_1(S_{C1})$	$E_2(S_{C2})$	$E_3(S_{C3})$
		$\prod E_1(S_{i1})$	$\prod E_2(S_{i2})$	$\prod E_3(S_{i3})$
		$E_1(\Sigma S_{i1})$	$E_2(\Sigma S_{i2})$	$E_3(\Sigma S_{i3})$
Total	$T=\Sigma V_i$	$T_1=\Sigma S_{i1}$	$T_2=\Sigma S_{i2}$	$T_3=\Sigma S_{i3}$

Interactive Proofs

- There are non-traditional methods of convincing others that something is true *without* writing down a proof.
- These methods can be used to convince others of the veracity of partial information about a secret.

Traditional Proofs

- I want to convince you that something is true.
- I write down a proof and give it to you.

Interactive Proofs

We engage in a dialogue at the conclusion of which you are convinced that my claim is true.

Proving Something is a Square

Suppose I want to convince you that Y is a square modulo N.

[There exists an X such that $Y = X^2 \bmod N$.]

Proving Something is a Square

Suppose I want to convince you that Y is a square modulo N.

[There exists an X such that $Y = X^2 \bmod N$.]

First approach: I give you X.

An Interactive Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ………… Y_{100}

An Interactive Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ………… Y_{100}

0 1 0 0 1 ………… 1

An Interactive Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ………… Y_{100}

0 1 0 0 1 ………… 1

$\sqrt{Y_1}$ $\sqrt{Y_3}$ $\sqrt{Y_4}$

An Interactive Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ………… Y_{100}

0 1 0 0 1 ………… 1

$\sqrt{Y_1}$ $\sqrt{Y_3}$ $\sqrt{Y_4}$

$\sqrt{(Y_2 \bullet Y)}$ $\sqrt{(Y_3 \bullet Y)}$ $\sqrt{(Y_{100} \bullet Y)}$

An Interactive Proof

In order for me to "fool" you, I would have to guess your exact challenge sequence.

The probability of my successfully convincing you that Y is a square when it is not is 2^{-100}.

This interactive proof is said to be "*zero-knowledge*" because the challenger received no information (beyond the proof of the claim) that it couldn't compute itself.

Proving Knowledge

Suppose that we share a public key consisting of a modulus N and an encryption exponent E and that I want to convince you that I have the corresponding decryption exponent D.

How can I do this?

Proving Knowledge

- I can give you my private key D.
- You can encrypt something for me and I decrypt it for you.
- You can encrypt something for me and I can engage in an interactive proof with you to show that I *can* decrypt it.

A Proof of Knowledge

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ………….. Y_{100}

A Proof of Knowledge

Y

Y_1	Y_2	Y_3	Y_4	Y_5	…………..	Y_{100}
0	1	0	0	1	…………..	1

A Proof of Knowledge

Y

Y_1	Y_2	Y_3	Y_4	Y_5	…………..	Y_{100}
0	1	0	0	1	…………..	1
Y_1^D		Y_3^D	Y_4^D			

A Proof of Knowledge

Y

Y_1	Y_2	Y_3	Y_4	Y_5	…………..	Y_{100}
0	1	0	0	1	…………..	1
Y_1^D		Y_3^D	Y_4^D			
	$(Y_2 \bullet Y)^D$			$(Y_5 \bullet Y)^D$		$(Y_{100} \bullet Y)^D$

A Proof of Knowledge

- By engaging in this proof, the prover has demonstrated its knowledge of Y^D – without revealing this value.
- If Y is generated by a challenger, this is compelling evidence that the prover posseses D.

Facts About Interactive Proofs

- Anything in PSPACE can be proven with an interactive proof.
- Anything in NP can be proven with a zero-knowledge interactive proof.

Facts about Interactive Proofs

- It is frequently possible to *simulate* the interaction by substituting a one-way function for the challenges of a verifier.

An Non-Interactive ZK Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$

An Non-Interactive ZK Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots \quad 1$

where the bit string is computed as
xxx = SHA-1(Y_1, Y_2,…, Y_{100})

An Non-Interactive ZK Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots \quad 1$

$\sqrt{Y_1} \quad \sqrt{Y_3} \quad \sqrt{Y_4}$

An Non-Interactive ZK Proof

Y

$Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5 \quad \cdots\cdots\cdots \quad Y_{100}$

$0 \quad 1 \quad 0 \quad 0 \quad 1 \quad \cdots\cdots\cdots \quad 1$

$\sqrt{Y_1} \quad \sqrt{(Y_2 \bullet Y)} \quad \sqrt{Y_3} \quad \sqrt{Y_4} \quad \sqrt{(Y_3 \bullet Y)} \quad \sqrt{(Y_{100} \bullet Y)}$

Elliptic Curve Cryptosystems

An elliptic curve

$$y^2 = x^3 + Ax + B$$

Elliptic Curves

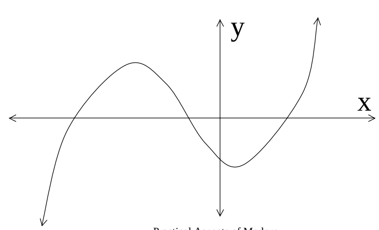
$$y^2 = x^3 + Ax + B$$

Elliptic Curves

$$y \;\; = x^3 + Ax + B$$

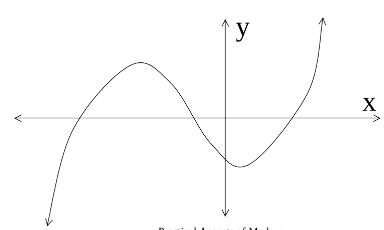
Elliptic Curves

$$y \;\; = x^3 + Ax + B$$



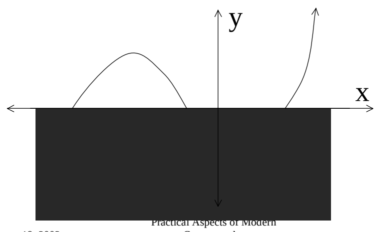
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



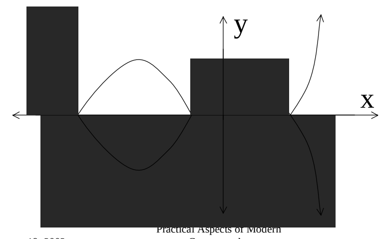
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



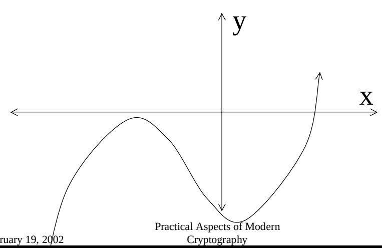
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



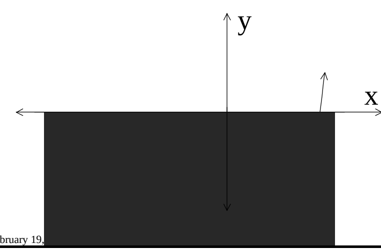
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



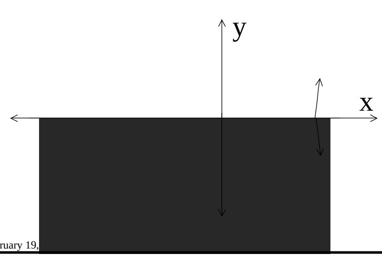
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



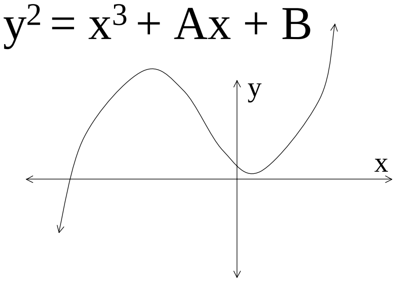
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



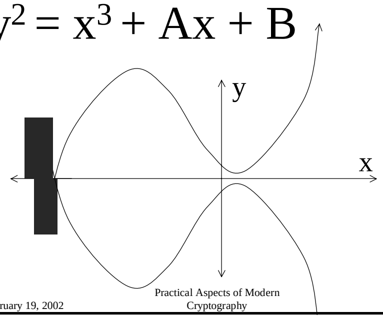
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



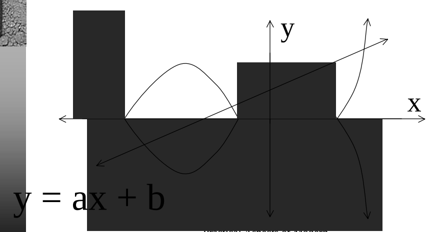
Elliptic Curves

$$y^2 = x^3 + Ax + B$$



Elliptic Curves Intersecting Lines

$$y^2 = x^3 + Ax + B$$



Elliptic Curves Intersecting Lines

Non-vertical Lines

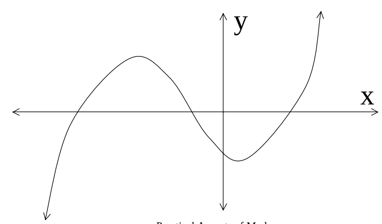
$$\begin{cases} y^2 = x^3 + Ax + B \\ y = ax + b \end{cases}$$

$$(ax + b)^2 = x^3 + Ax + B$$

$$x^3 + A'x^2 + B'x + C' = 0$$

Elliptic Curves Intersecting Lines

$$x^3 + A'x^2 + B'x + C' = 0$$



Elliptic Curves Intersecting Lines

Non-vertical Lines

- 1 intersection point (typical case)
- 2 intersection points (tangent case)
- 3 intersection points (typical case)

Elliptic Curves Intersecting Lines

Vertical Lines

$$\begin{cases} y^2 = x^3 + Ax + B \\ x = c \end{cases}$$

$$y^2 = c^3 + Ac + B$$

$$y^2 = C$$

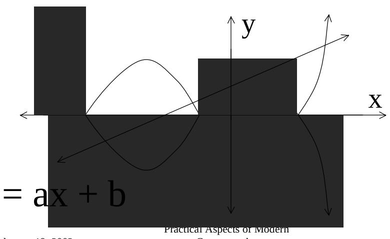
Elliptic Curves Intersecting Lines

Vertical Lines

- 0 intersection point (typical case)
- 1 intersection points (tangent case)
- 2 intersection points (typical case)

Elliptic Groups

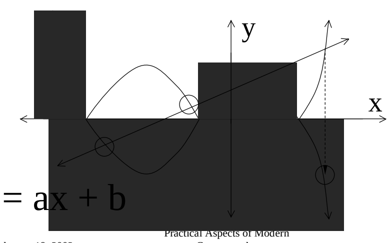
$$y^2 = x^3 + Ax + B$$



$y = ax + b$

Elliptic Groups

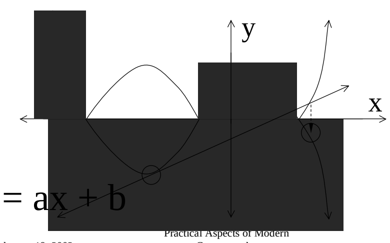
$$y^2 = x^3 + Ax + B$$



$y = ax + b$

Elliptic Groups

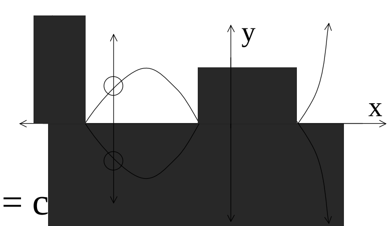
$$y^2 = x^3 + Ax + B$$



$y = ax + b$

Elliptic Groups

$$y^2 = x^3 + Ax + B$$



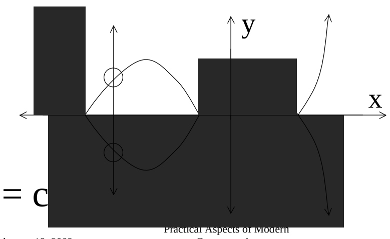
$x = c$

Elliptic Groups

- Add an “artificial” point I to handle the vertical line case.
- This point I also serves as the group identity value.

Elliptic Groups

$$y^2 = x^3 + Ax + B$$



x = c

Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

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

when $x_1 \neq x_2$

Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = (x_3,y_3)$$

$$x_3 = ((3x_1^2+A)/(2y_1))^2 - 2x_1$$
$$y_3 = -y_1 + ((3x_1^2+A)/(2y_1))\ (x_1 - x_3)$$

when $x_1 = x_2$ and $y_1 = y_2 \neq 0$

Elliptic Groups

$$(x_1,y_1) \times (x_2,y_2) = I$$
when $x_1 = x_2$ but $y_1 \neq y_2$ or $y_1 = y_2 = 0$

$$(x_1,y_1) \times I = (x_1,y_1) = I \times (x_1,y_1)$$

$$I \times I = I$$

The Fundamental Equation

$$Z=Y^X \bmod N$$

The Fundamental Equation

$$Z=Y^X \text{ in } E_p(A,B)$$

The Fundamental Equation

$$Z=Y^X \text{ in } E_p(A,B)$$

When Z is unknown, it can be efficiently computed by repeated squaring.

The Fundamental Equation

$$Z=Y^X \text{ in } E_p(A,B)$$

When X is unknown, this version of the discrete logarithm is believed to be quite hard to solve.

The Fundamental Equation

$$Z=Y^X \text{ in } E_p(A,B)$$

When Y is unknown, it *can* be efficiently computed by "sophisticated" means.

Diffie-Hellman Key Exchange

Alice	Bob
♦ Randomly select a large integer *a* and send A = Y^a mod N.	♦ Randomly select a large integer *b* and send B = Y^b mod N.
♦ Compute the key K = B^a mod N.	♦ Compute the key K = A^b mod N.

$$B^a = Y^{ba} = Y^{ab} = A^b$$

Diffie-Hellman Key Exchange

Alice	Bob
♦ Randomly select a large integer *a* and send A = Y^a in E_p.	♦ Randomly select a large integer *b* and send B = Y^b in E_p.
♦ Compute the key K = B^a in E_p.	♦ Compute the key K = A^b in E_p.

$$B^a = Y^{ba} = Y^{ab} = A^b$$

Why use Elliptic Curves?

- The best *currently known* algorithm for EC discrete logarithms would take about as long to find a 160-bit EC discrete log as the best *currently known* algorithm for integer discrete logarithms would take to find a 1024-bit discrete log.
- 160-bit EC algorithms are somewhat faster and use shorter keys than 1024-bit "traditional" algorithms.

Why *not* use Elliptic Curves?

- EC discrete logarithms have been studied far less than integer discrete logarithms.
- Results have shown that a fundamental break in integer discrete logs would also yield a fundamental break in EC discrete logs, although the reverse may not be true.
- Basic EC operations are more cumbersome than integer operations, so EC is only faster if the keys are *much* smaller.