Practical Aspects of Modern Cryptography

Josh Benaloh
Brian LaMacchia
John Manferdelli

Fun with Public-Key

Tonight we'll …

- Introduce some basic tools of public-key crypto
- Combine the tools to create more powerful tools
- Apply these tools to a grand application: doing elections "right"

Interactive Proofs and Zero Knowledge

- 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.

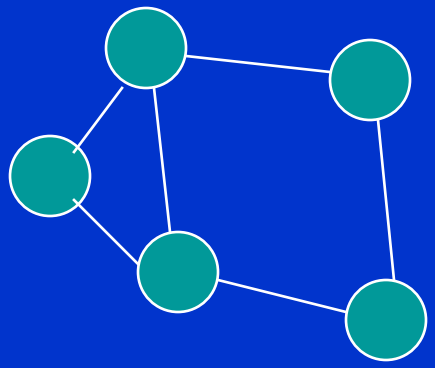
Traditional Proofs

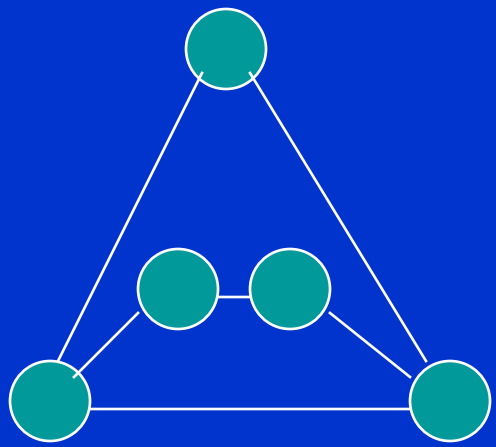
- I want to convince you that something is true.
- I write down a proof and give it to you.

Interactive Proofs

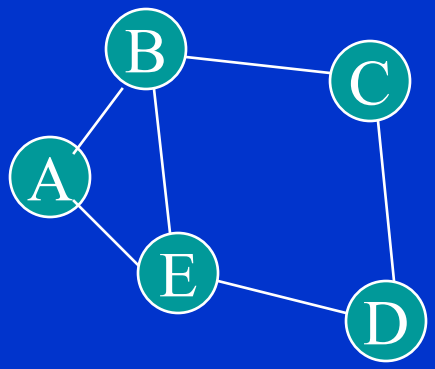
We engage in a dialogue at the conclusion of which you are convinced that my claim is true.

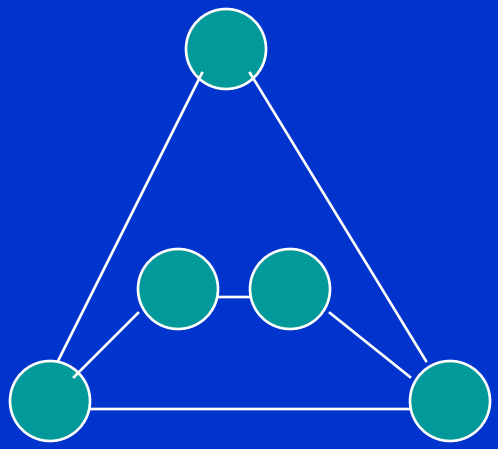
Graph Isomorphism



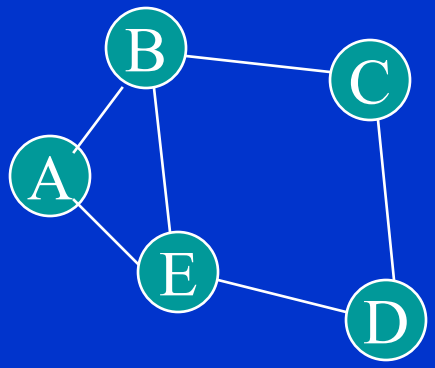


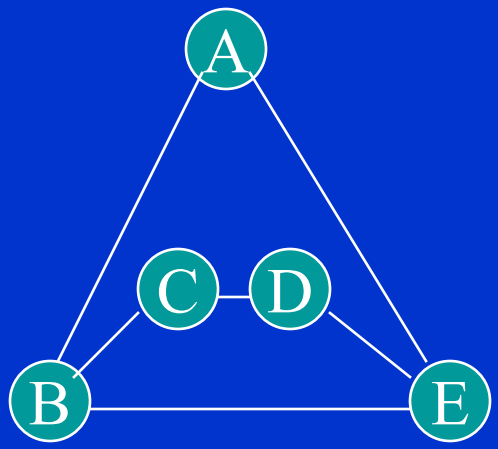
Graph Isomorphism





Graph Isomorphism





Zero-Knowledge Interactive Proof

G_1 G_2

ZKIP of Graph Isomorphism

Generate, say, 100 additional graphs isomorphic to G_1 (and therefore also isomorphic to G_2).

ZKIP of Graph Isomorphism

H_1

H_2

H_3

G_1 G_2

H_{100}

ZKIP of Graph Isomorphism

- Accept a single bit challenge “L/R” for each of the 100 additional graphs.
- Display the indicated isomorphism for each of the additional graphs

ZKIP of Graph Isomorphism

H_1

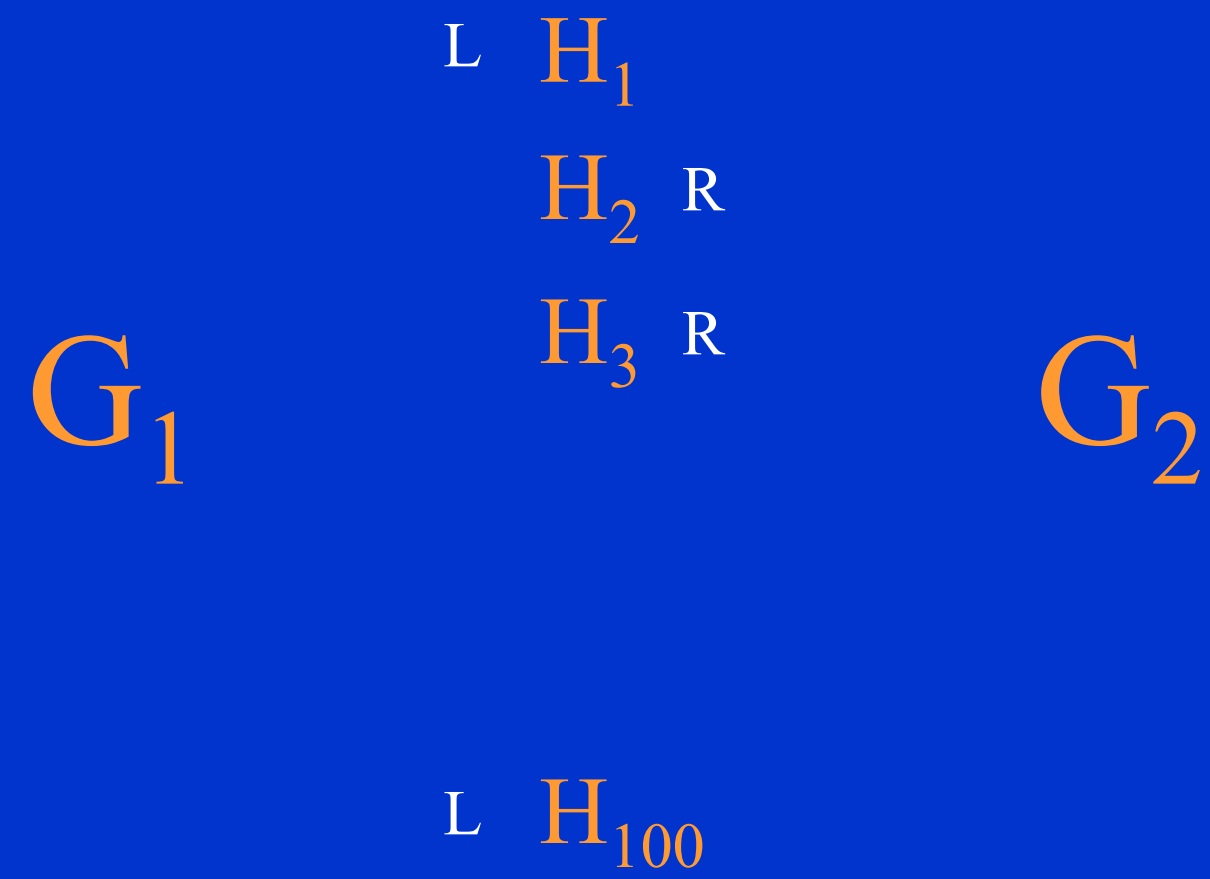
H_2

H_3

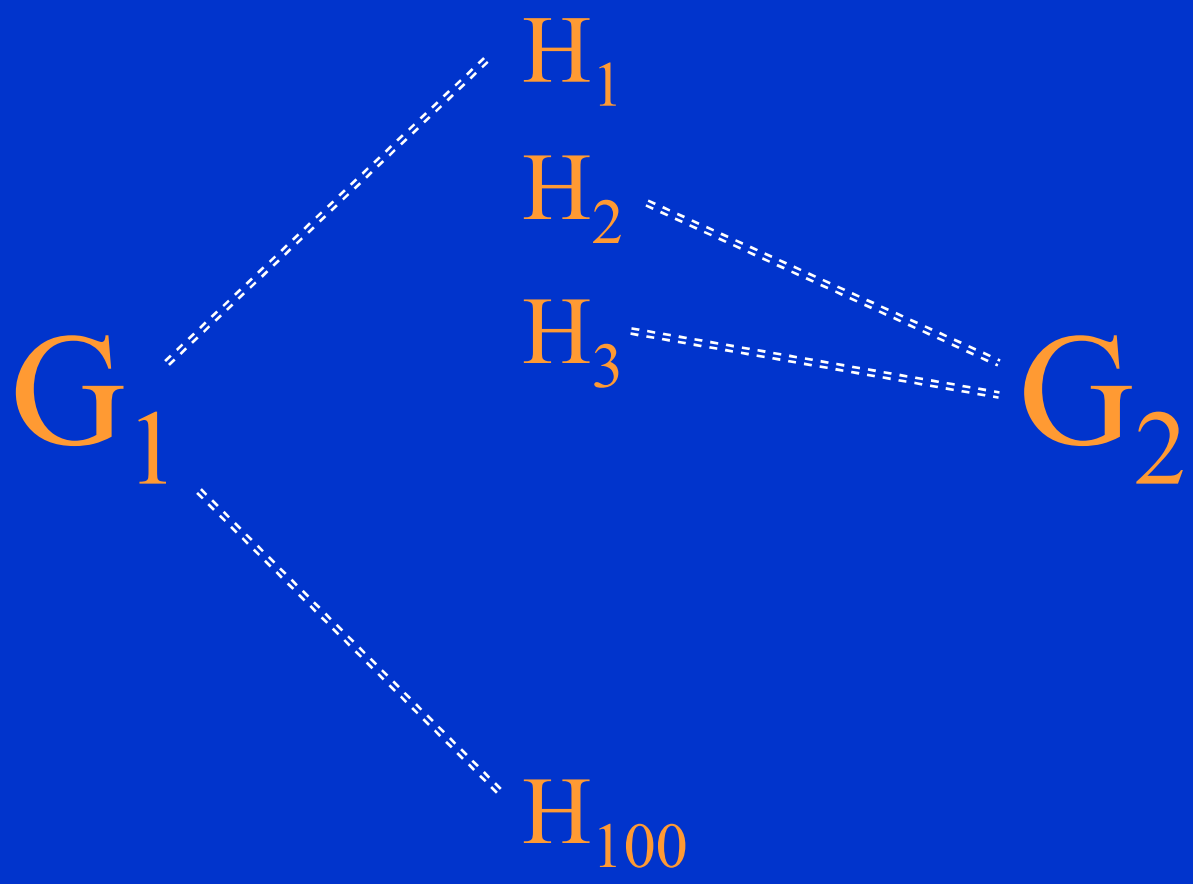
G_1 G_2

H_{100}

ZKIP of Graph Isomorphism



ZKIP of Graph Isomorphism



ZKIP of Graph Isomorphism

- If graphs G_1 and G_2 were *not* isomorphic, then the "prover" would not be able to show any additional graph to be isomorphic to *both* G_1 and G_2.
- A successful false proof would require the prover to guess all 100 challenges in advance: probability 1 in 2^{100}.

Fiat-Shamir Heuristic

- Instead of challenge bits being externally generated, they can be produced by applying a one-way hash function to the full set of additional graphs.
- This allows a ZKIP to be “published” without need for interaction.

ZKIP of Graph Non-Isomorphism

G_1 G_2

ZKIP of Graph Non-Isomorphism

- A verifier can generate 100 additional graphs, each isomorphic to one of G_1 and G_2 , and present them to the prover.
- The prover can then demonstrate that the graphs are not isomorphic by identifying which of G_1 and G_2 each additional graph is isomorphic to.

ZKIP of Graph Non-Isomorphism

G_1 G_2

ZKIP of Graph Non-Isomorphism

H_1

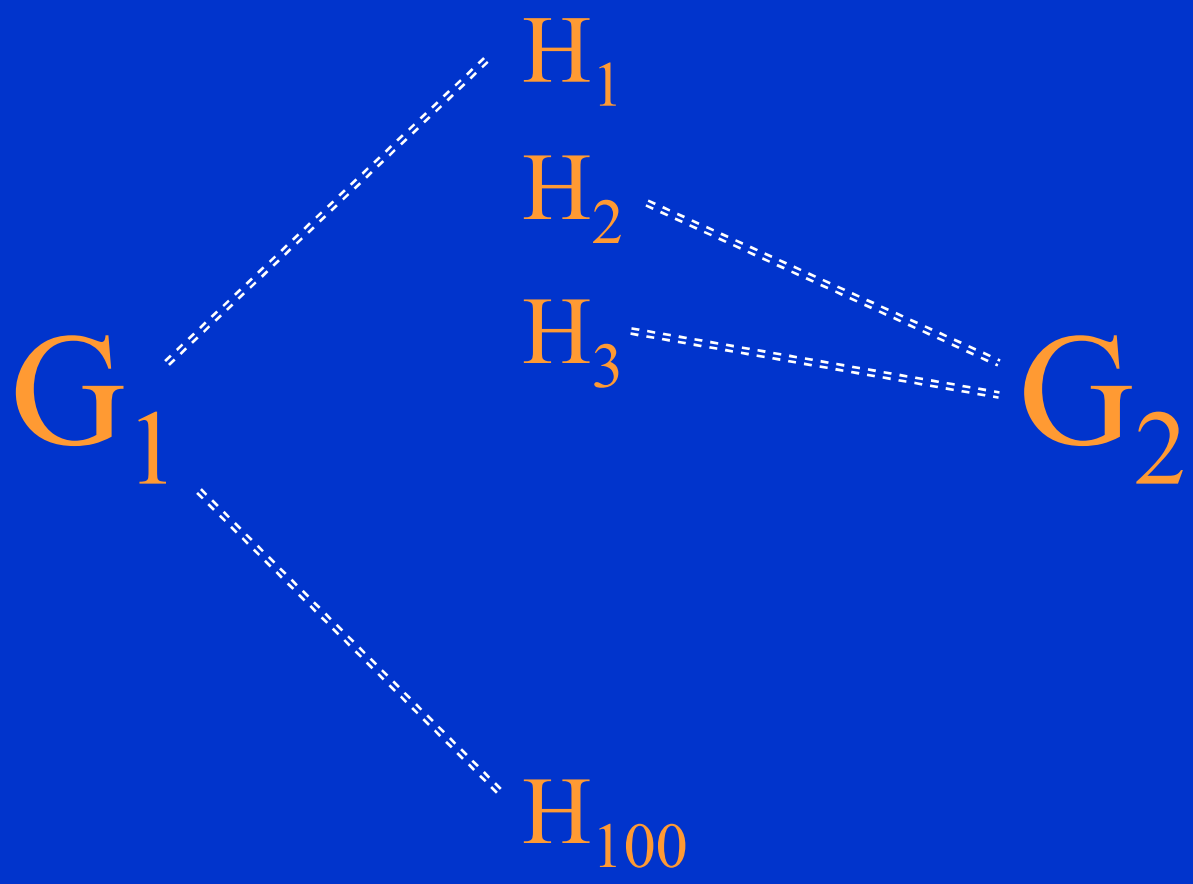
H_2

H_3

G_1 G_2

H_{100}

ZKIP of Graph Non-Isomorphism



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

$$Y$$

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

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

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

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

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

Proving Knowledge

- I can give you my private key D.

Proving Knowledge

- I can give you my private key D.
- You can encrypt something for me and I decrypt it for you.

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 possesses 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.

Applying Fiat-Shamir

Once again, the verifier challenges can be simulated by the use of a one-way function to generate the challenge bits.

An Non-Interactive ZK Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ..... Y_{100}

An Non-Interactive ZK Proof

Y

Y_1 Y_2 Y_3 Y_4 Y_5 ..... Y_{100}

0 1 0 0 1 1

where the bit string is computed as

xxx = SHA-1(Y_1, Y_2,…, Y_{100})

An Non-Interactive ZK Proof

$$Y$$

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

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

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

An Non-Interactive ZK Proof

$$Y$$

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

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

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

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 $z \in Z_n = \{0,1,\ldots,m\text{-}1\}$ 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

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

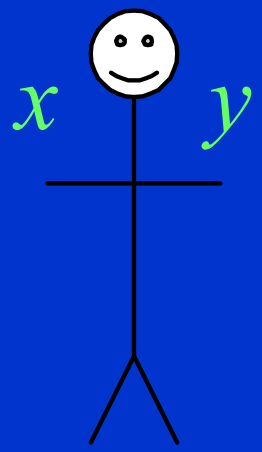
Secret Sharing – AND

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

Secret Sharing – AND

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

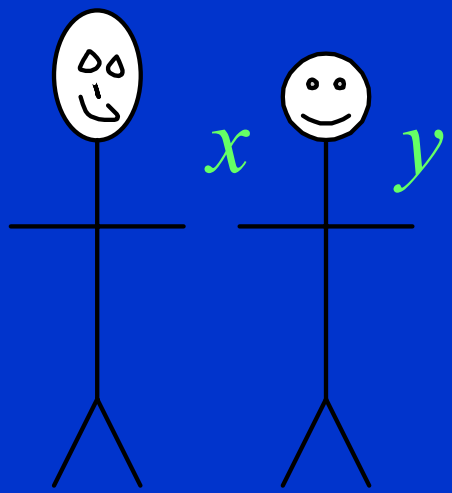
Me



Secret Sharing – AND

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

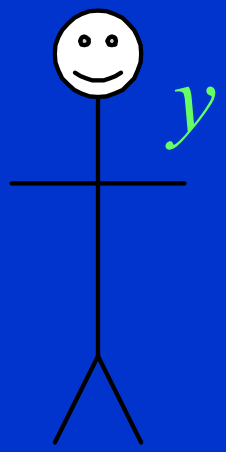
Alice Me



Secret Sharing – AND

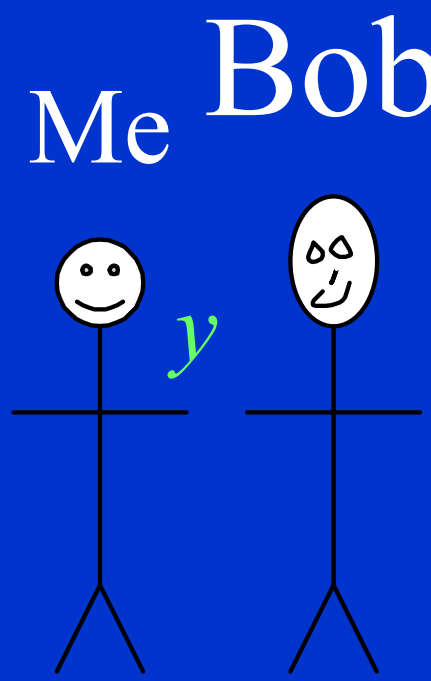
The secret value is $z = (x + y) \bmod m$.

Me



Secret Sharing – AND

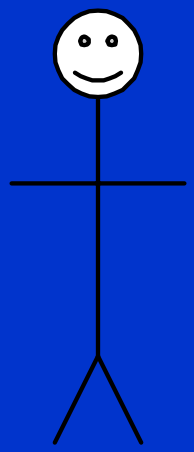
The secret value is $z = (x + y) \bmod m$.



Secret Sharing – AND

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

Me



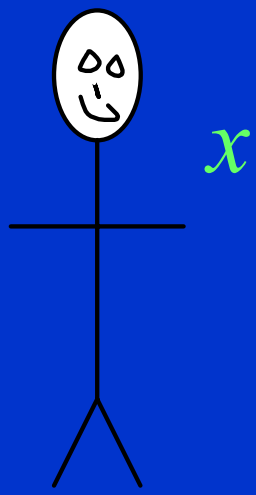
Secret Sharing – AND

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

Secret Sharing – AND

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

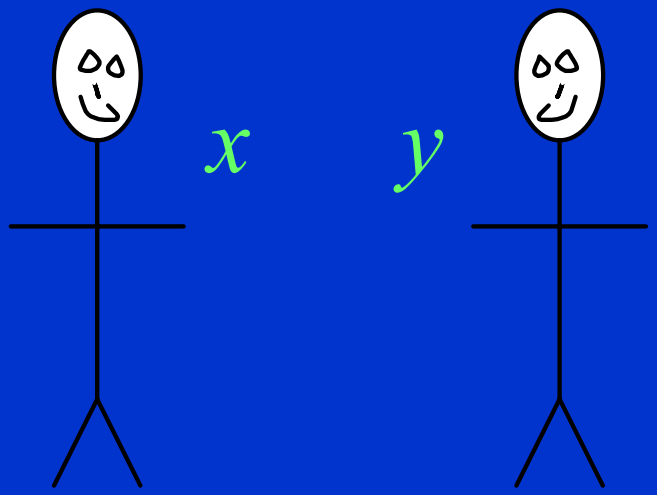
Alice



Secret Sharing – AND

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

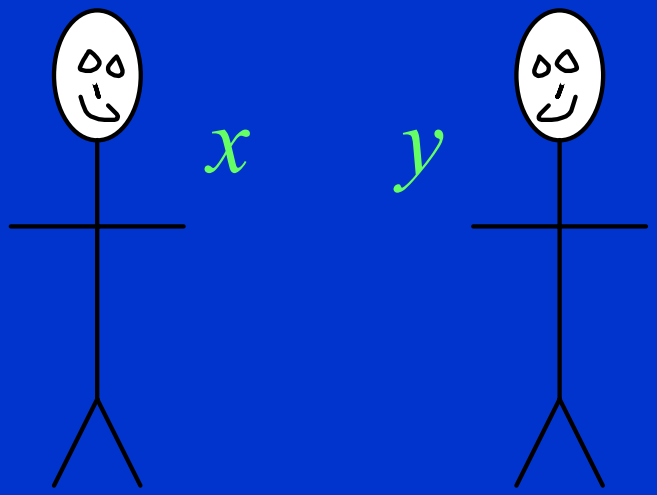
Alice Bob



Secret Sharing – AND

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

Alice Bob



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 $0 \leq s_i < m$.

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.

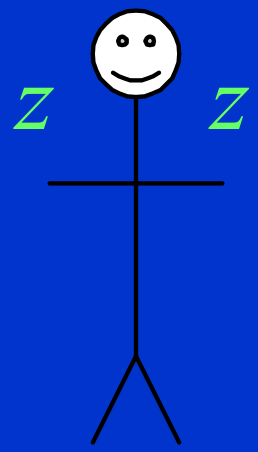
Secret Sharing – OR

The secret value is z.

Secret Sharing – OR

The secret value is z.

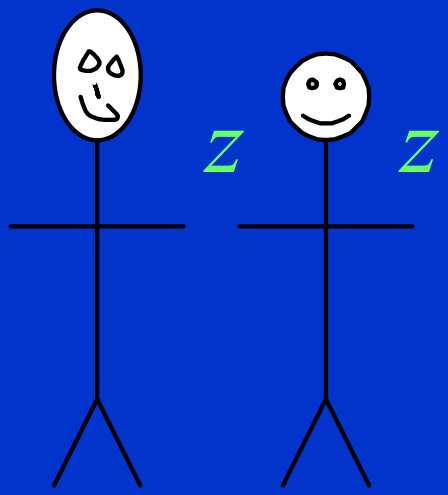
Me



Secret Sharing – OR

The secret value is z.

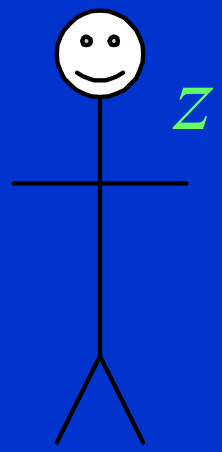
Alice Me



Secret Sharing – OR

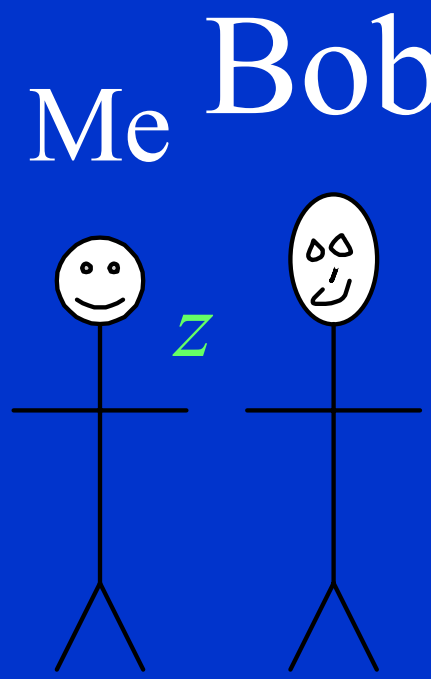
The secret value is z.

Me



Secret Sharing – OR

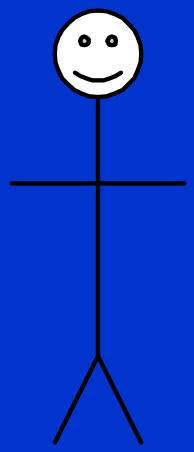
The secret value is z.



Secret Sharing – OR

The secret value is z.

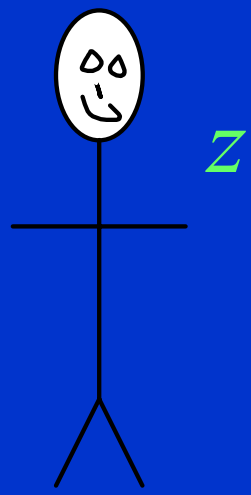
Me



Secret Sharing – OR

The secret value is z.

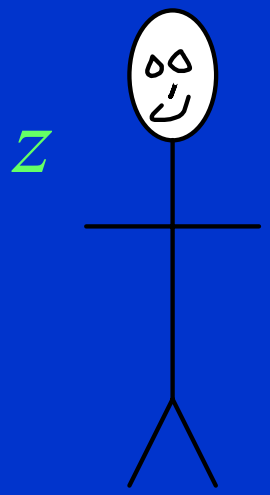
Alice



Secret Sharing – OR

The secret value is z.

Bob



Secret Sharing – OR

This case also generalizes easily to more than two shareholders.

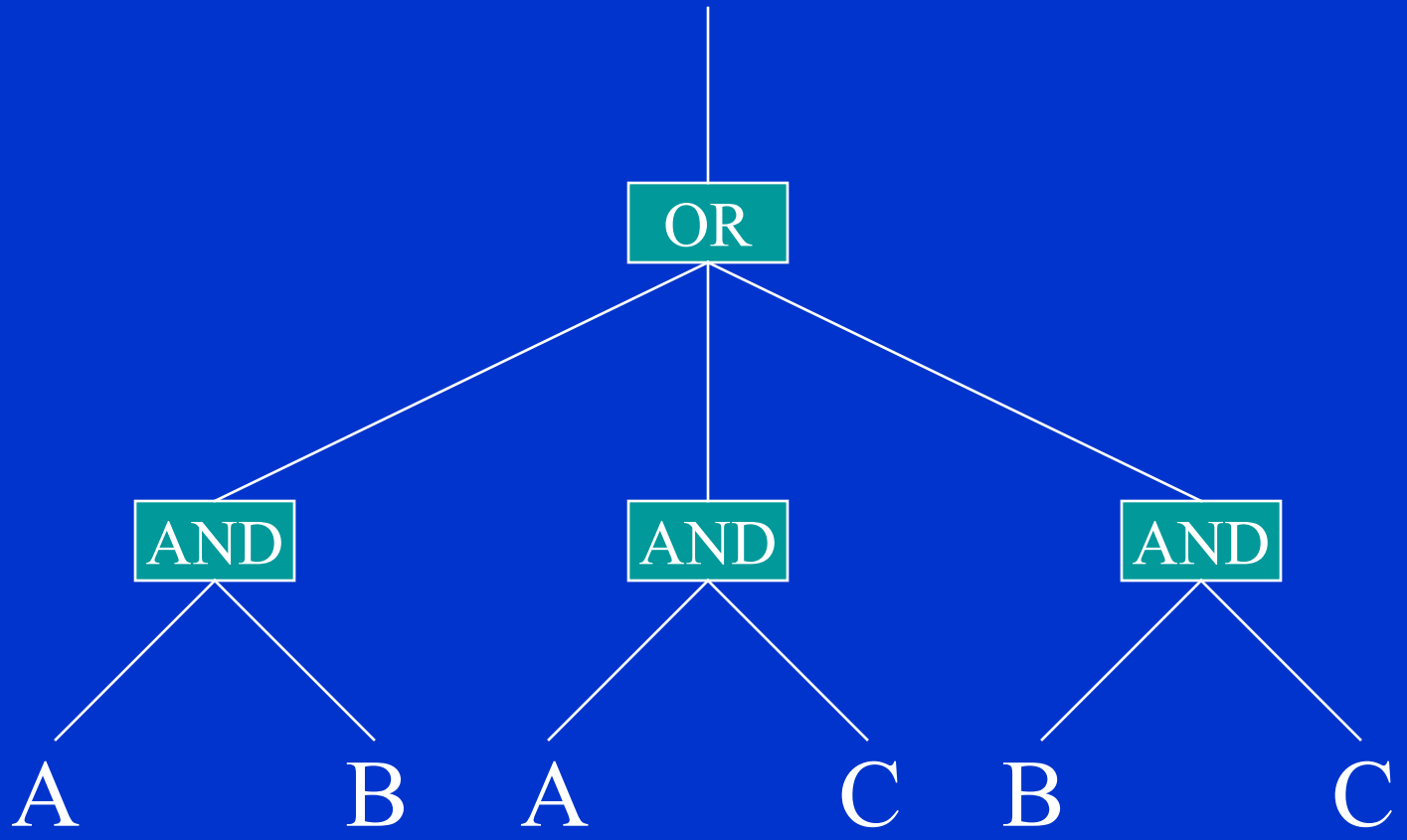
Secret Sharing

More complex *access structures* …

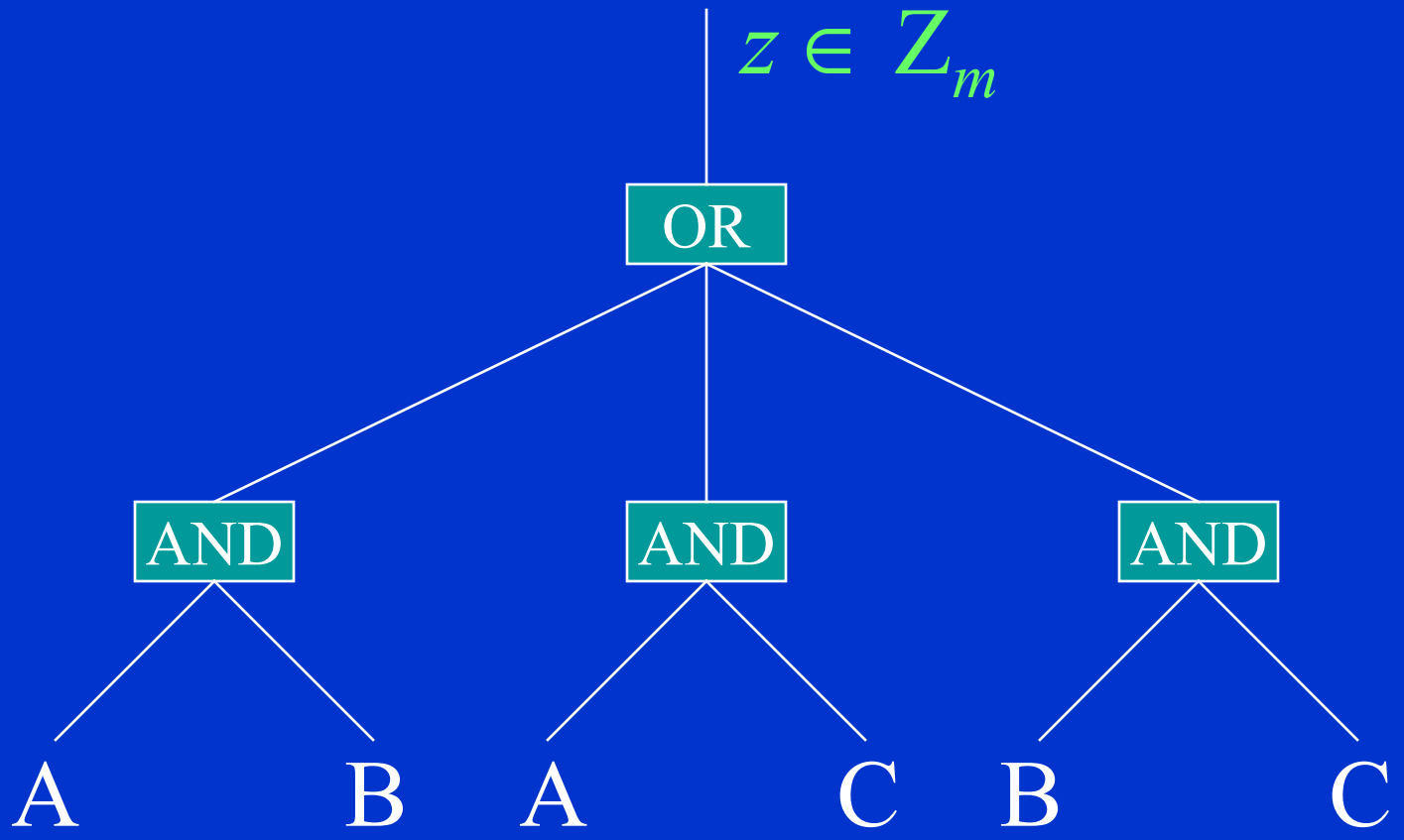
I want to share secret value z amongst Alice, Bob, and Carol such that any two of the three can reconstruct z.

$$S = (\mathrm{A} \wedge \mathrm{B}) \vee (\mathrm{A} \wedge \mathrm{C}) \vee (\mathrm{B} \wedge \mathrm{C})$$

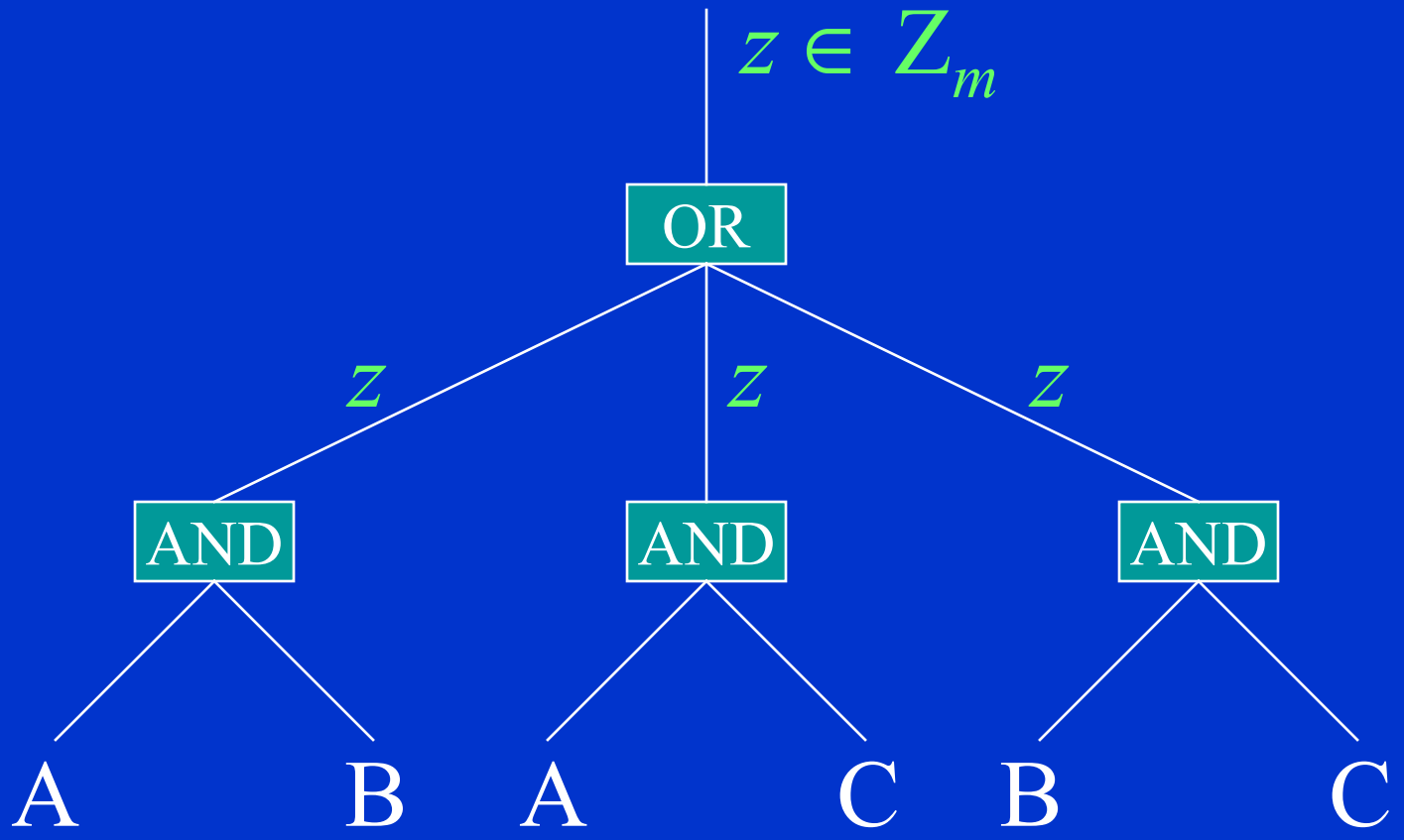
Secret Sharing



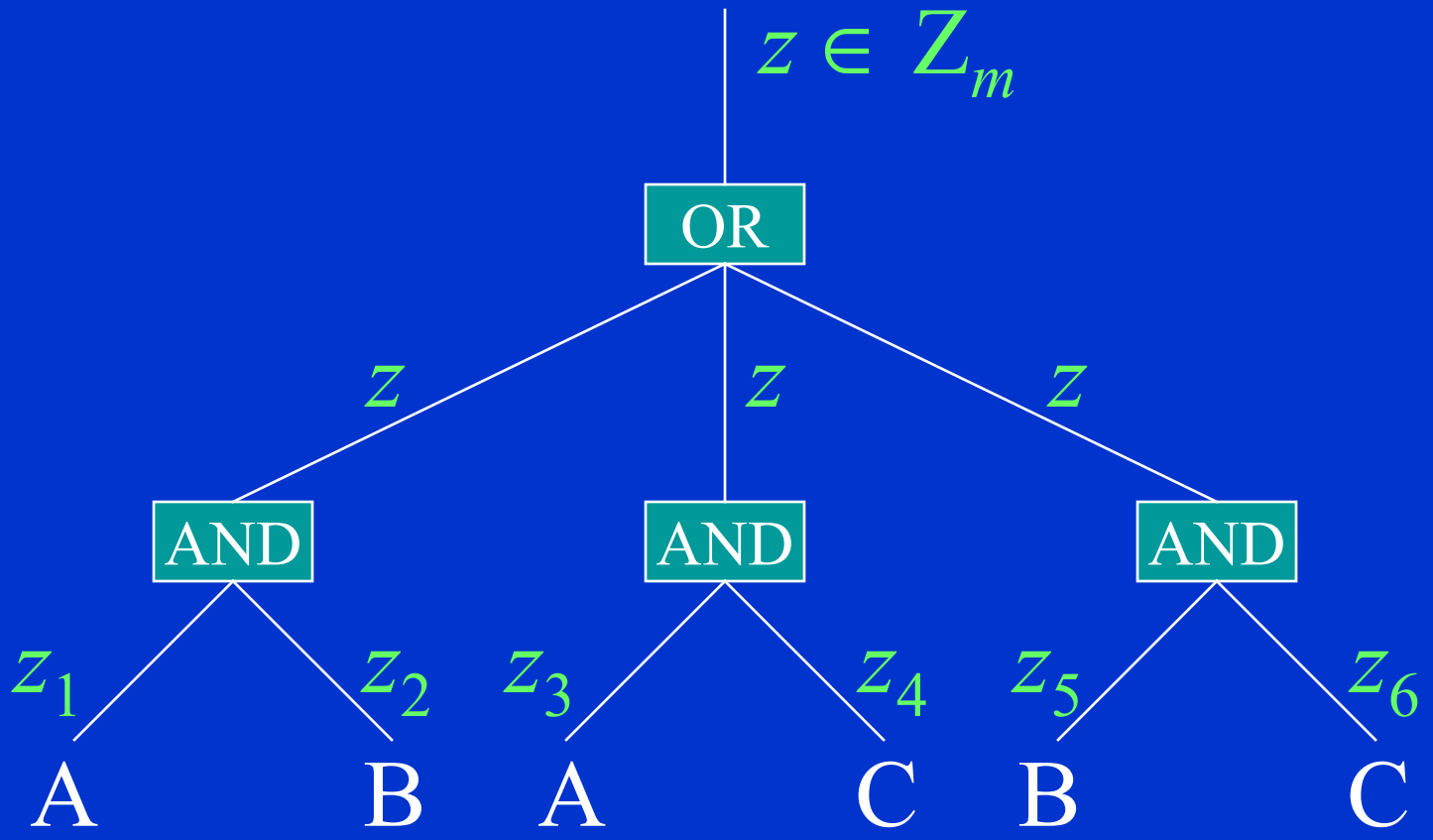
Secret Sharing



Secret Sharing



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 $Z_p = \{0,1,\ldots,p\text{-}1\}$ 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 = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9

Shamir's Threshold Scheme

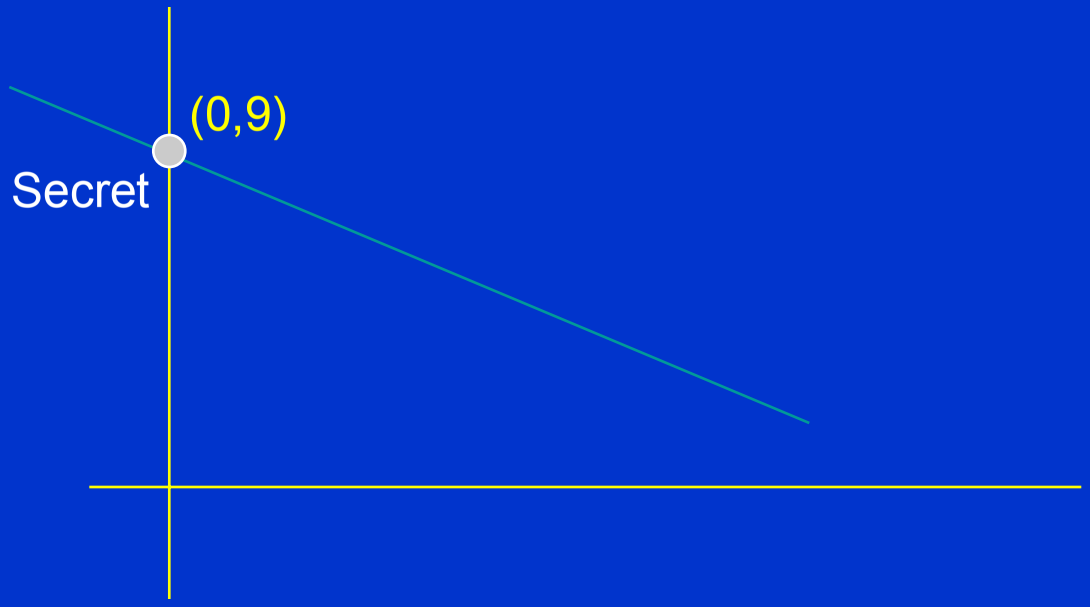
The threshold 2 case:

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

Shamir's Threshold Scheme

The threshold 2 case:

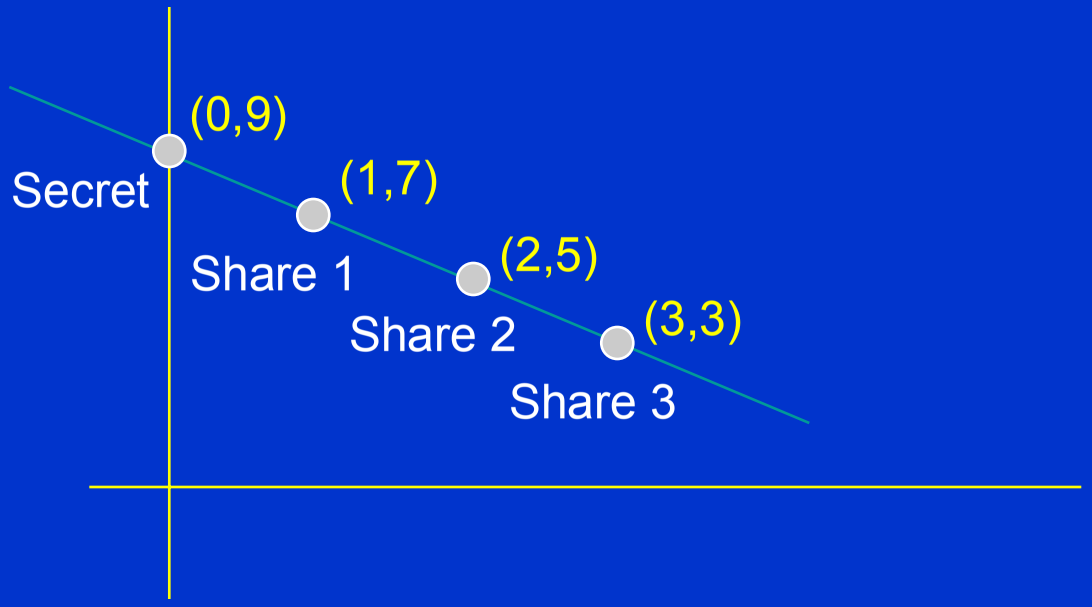
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

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



Shamir's Threshold Scheme

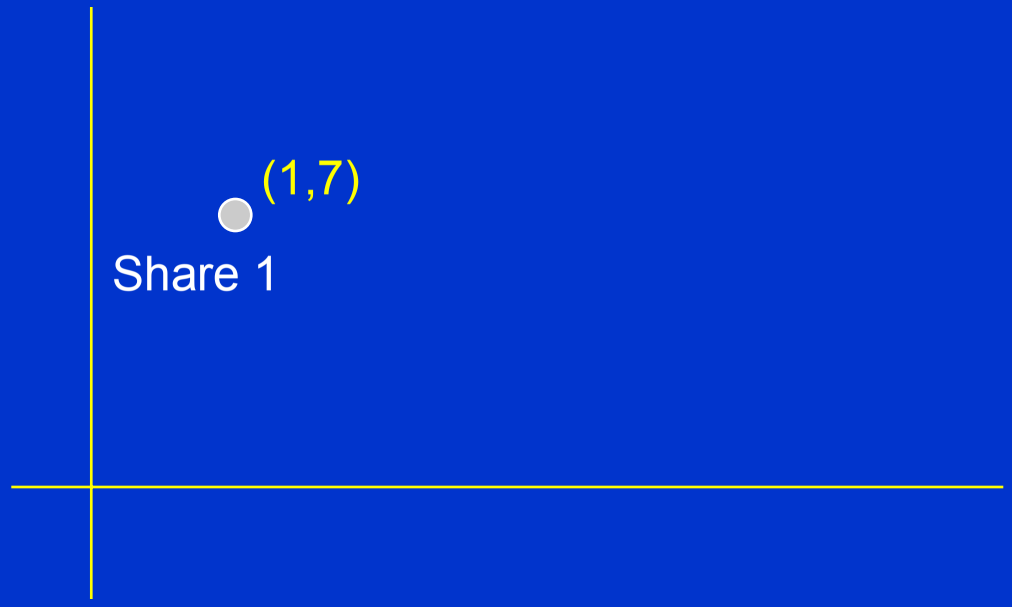
The threshold 2 case:

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

Shamir's Threshold Scheme

The threshold 2 case:

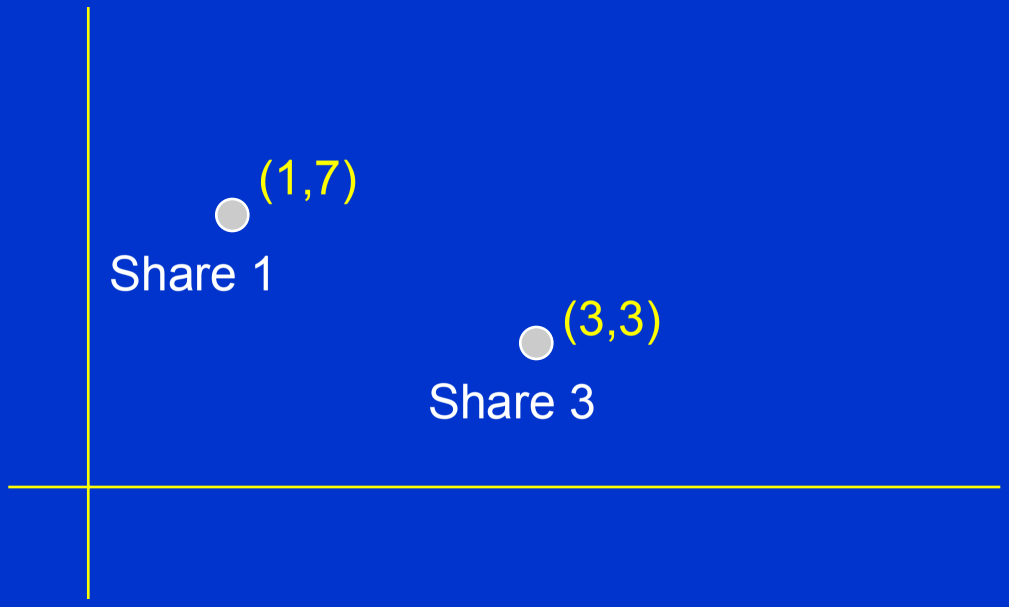
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

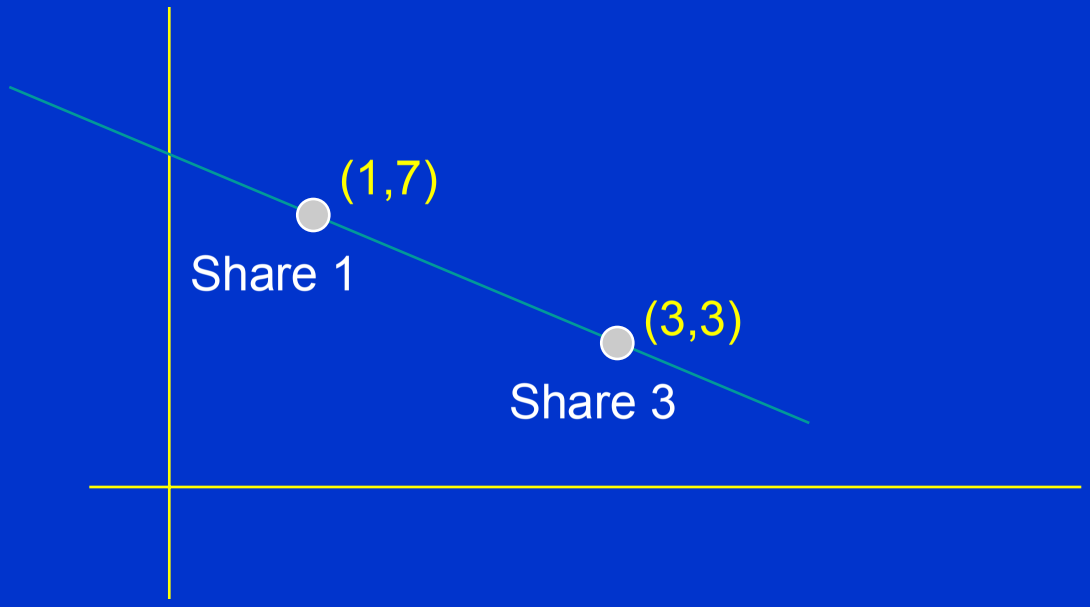
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

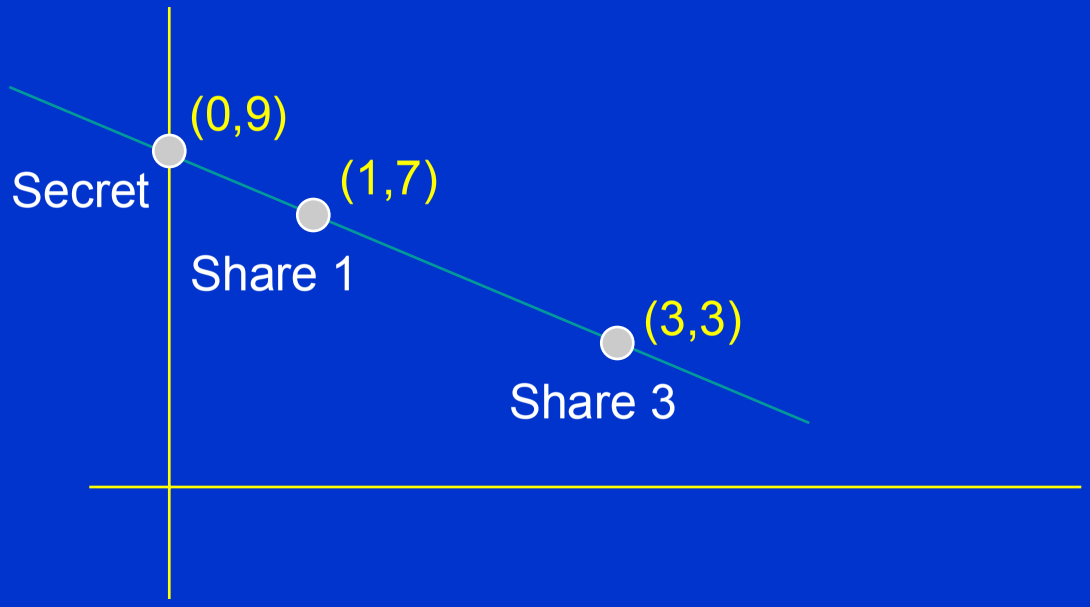
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$, Secret = 9



Shamir's Threshold Scheme

The threshold 2 case:

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



Shamir's Threshold Scheme

The threshold 2 case:

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

Shamir's Threshold Scheme

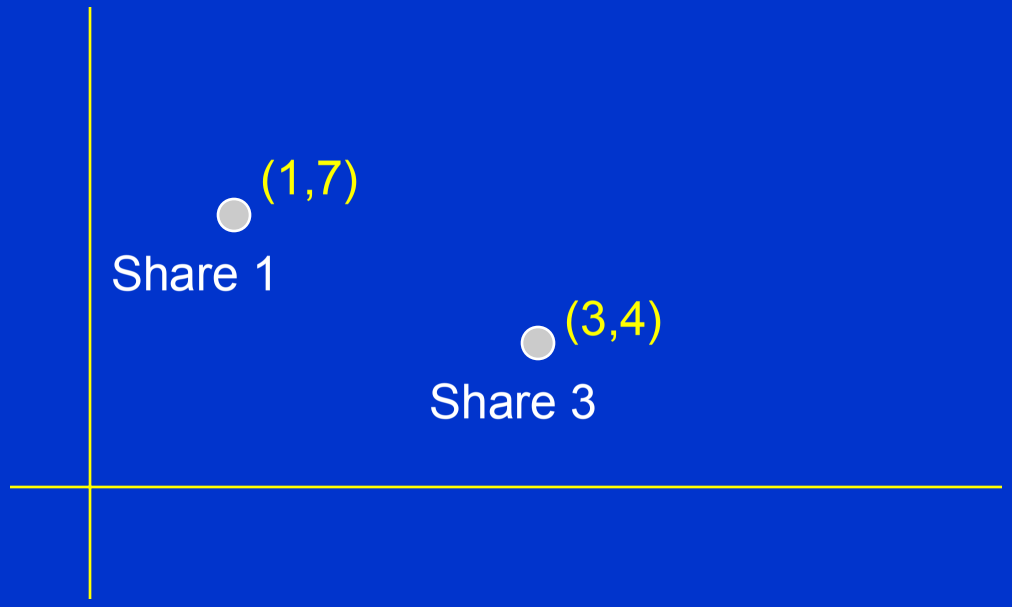
The threshold 2 case:

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

Shamir's Threshold Scheme

The threshold 2 case:

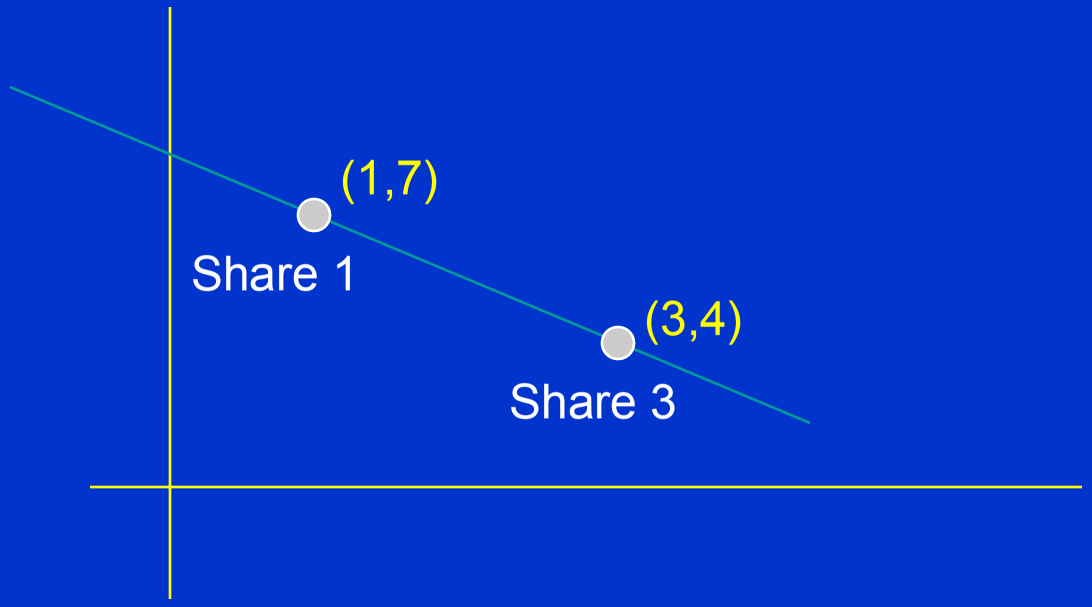
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$



Shamir's Threshold Scheme

The threshold 2 case:

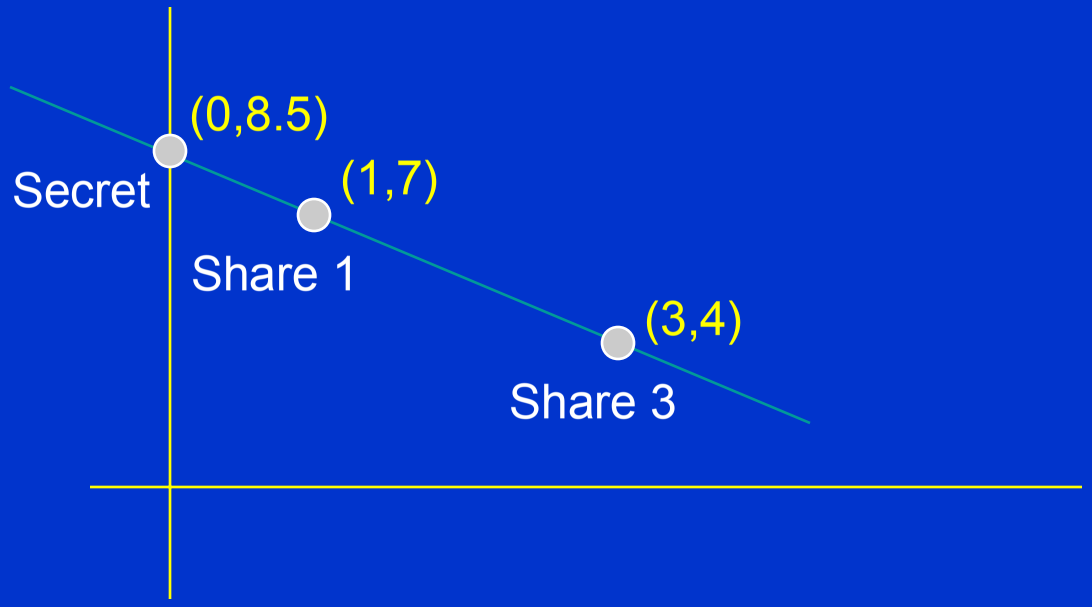
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$



Shamir's Threshold Scheme

The threshold 2 case:

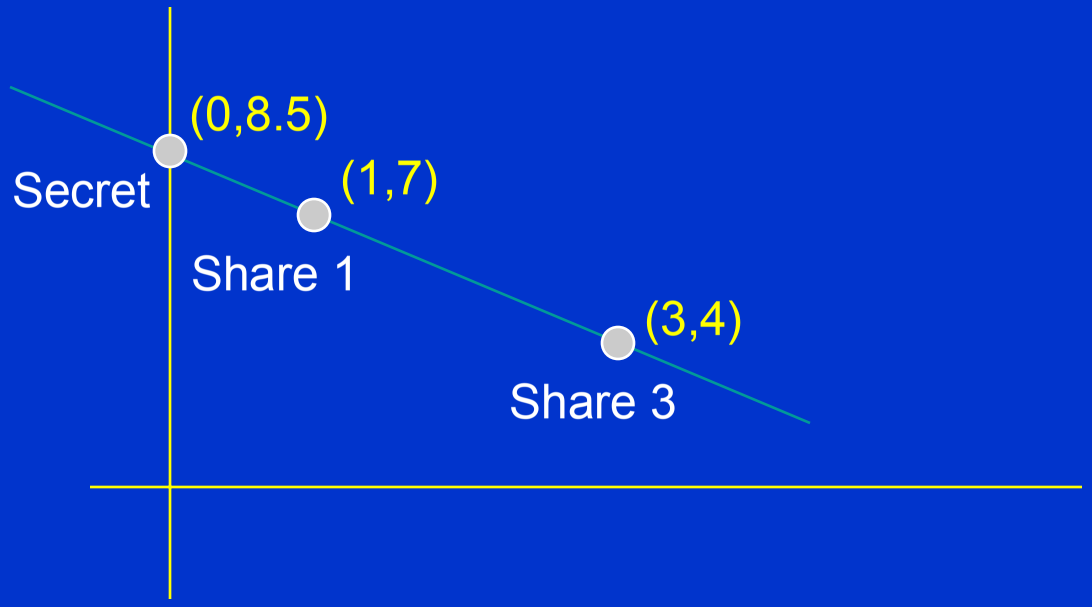
Example: Range = $Z_{11} = \{0,1,\ldots,10\}$



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 coefficients 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: $P_1(1) + P_2(1), P_1(2) + P_2(2), \ldots,$
$P_1(n) + P_2(n)$

Threshold Encryption

I want to encrypt a secret message M for a set of *n* recipients such that

- any *k* of the *n* recipients can uniquely decrypt the secret message M,
- but any set of fewer than *k* recipients has *no information whatsoever* about the secret message M.

Recall Diffie-Hellman

Alice

- Randomly select a large integer a and send $A = g^a \bmod p$.
- Compute the key $K = B^a \bmod p$.

Bob

- Randomly select a large integer b and send $B = g^b \bmod p$.
- Compute the key $K = A^b \bmod p$.

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

ElGamal Encryption

- Alice select a large random private key a and computes an associated public key $A = g^a \bmod p$.
- To send a message M to Alice, Bob selects a random value r and computes the pair $(X,Y) = (A^r M \bmod p, g^r \bmod p)$.
- To decrypt, Alice computes $X/Y^a \bmod p = A^r M/g^{ra} \bmod p = M$.

ElGamal Re-Encryption

If $A = g^a \bmod p$ is a public key and the pair

$$(X,Y) = (A^r M \bmod p,\ g^r \bmod p)$$

is an encryption of message M, then for any value c, the pair

$$(A^c X, g^c Y) = (A^{c+r} M \bmod p,\ g^{c+r} \bmod p)$$

is an encryption of the same message M, for any value c.

Group ElGamal Encryption

- Each recipient selects a large random private key a_i and computes an associated public key $A_i = g^{a_i} \bmod p$.
- The group key is $A = \prod A_i \bmod p = g^{\Sigma a_i} \bmod p$.
- To send a message M to the group, Bob selects a random value r and computes the pair $(X,Y) = (A^r M \bmod p, g^r \bmod p)$.
- To decrypt, each group member computes $Y_i = Y^{a_i} \bmod p$. The message $M = X/\prod Y_i \bmod p$.

Threshold Encryption (ElGamal)

- Each recipient selects k large random secret coefficients $a_{i,0}, a_{i,1}, \ldots a_{i,k-1}$ and forms the polynomial

$$P_i(x) = a_{i,k-1}x^{k-1} + a_{i,k-2}x^{k-2} + a_{i,1}x + a_{i,0}$$

- Each polynomial $P_i(x)$ is then verifiably shared with the other recipients by distributing each $g^{a_{i,j}}$.
- The joint (threshold) public key is $\prod g^{a_{i,0}}$.

Threshold Encryption (ElGamal)

- The joint (threshold) public key is $\prod g^{a_{i,0}}$.

Can Secure Election Systems be Built on Software?

Disclaimer:

Any opinions expressed here regarding elections are my own and do not necessary represent those of the Microsoft Corporation or any subsidiary or partner thereof.

A Claim:

A Claim:

I know software …

A Claim:

I know software …

I've worked with software …

A Claim:

I know software …

I've worked with software …

You cannot trust software.

More specifically …

There are a million ways to tamper with software:

- Insider attacks
- Exploitation of bugs and vulnerabilities
- Configuration errors
- etc.

How can one trust an election to software?

A Web-Based Election

A Web-Based Election

- Voters post their names and votes to a public web site.

A Web-Based Election

- Voters post their names and votes to a public web site.
- Anyone who cares to do so can
  - Check that their own votes are correctly posted
  - Check that other voters are legitimate
  - Check that the totals are correct

But wait …

But wait …

This isn't a *secret-ballot* election.

But wait …

This isn't a *secret-ballot* election.

Quite true, but it's enough to falsify the previous arguments.

Privacy

Privacy

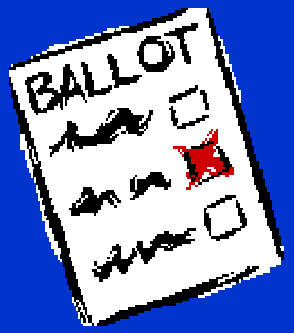
- The only ingredient missing from this “toy” web-based election is privacy – and the things which flow from privacy (e.g. protection from coercion).

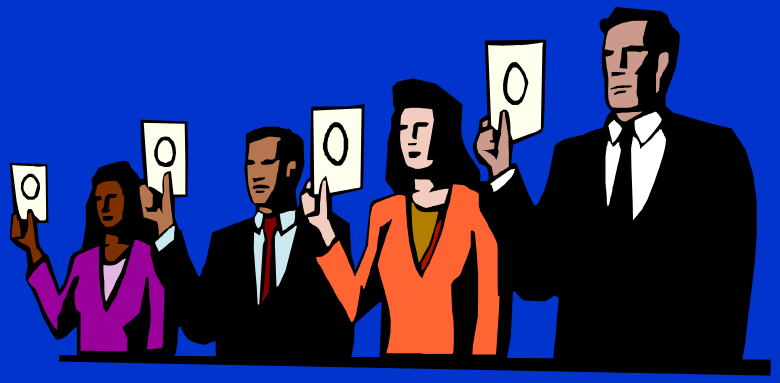
Privacy

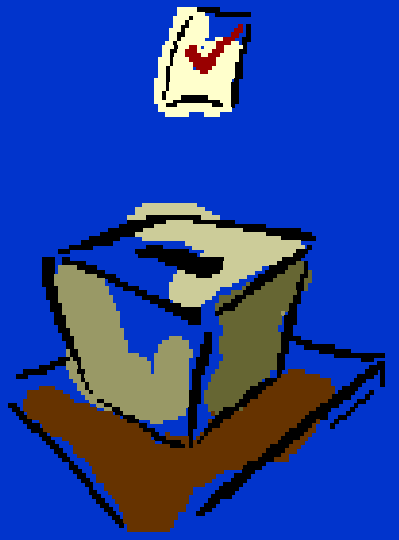
- The only ingredient missing from this "toy" web-based election is privacy – and the things which flow from privacy (e.g. protection from coercion).
- Performing tasks while preserving privacy is the bailiwick of cryptography.

Understanding Elections

So, you want to hold an election …







Some Principles of Election Protocols

- Privacy
- Verifiability
- Robustness
- Coercibility

Privacy

- Only one voter?
- A unanimous tally?
- Unanimous less one?

Verifiability

- By single trusted party?
- By trusted committee?
- By each voter?
- By observers?

Robustness

- Against faulty/malicious voters?
- Against faulty/malicious officials?
- At what cost to privacy?

Coercibility

- When?
- By whom? (voter, official, or observer)
- Where?
- Free-form ballots?

Current Election Methods

Current Election Methods

- Currently deployed touch-screen systems have good usability properties but no substantive verifiability.

Current Election Methods

- Currently deployed touch-screen systems have good usability properties but no substantive verifiability.
- Paper-based systems offer some verifiability, but voters can only track their votes to a limited extent.

Current Election Methods

- Currently deployed touch-screen systems have good usability properties but no substantive verifiability.
- Paper-based systems offer some verifiability, but voters can only track their votes to a limited extent.
- Perhaps we can do even better.

Cryptographic Verifiability

- With well-built paper-based systems, voters can ensure that their intended votes went into a locked ballot box but must depend upon officials and procedures to ensure that their votes are included in the tally.

Cryptographic Verifiability

- With well-b stems, voters can ensure t otes went into a locke st depend upon offici o ensure that their votes a ally.

Cryptographic Verifiability

- With well-built paper-based systems, voters can ensure that their intended votes went into a locked ballot box but must depend upon officials and procedures to ensure that their votes are included in the tally.
- Cryptographic methods can give voters complete confidence that their intended votes were properly included in the tally.

Cryptographic Voting Systems

There are many approaches to cryptographic voting, but the primary options can be divided into two phases.

1. Voters transform their intentions into *encrypted* ballots and post their (named) ballots on a public list.
2. The list of encrypted votes is publicly processed to produce a tally *and* a proof that the tally is correct.

The Encryption Phase

Turning your intentions into an encrypted ballot should be easy – no?

- You can use your own machine.
- You can use any machine you trust.
- You can use a dedicated device.

Researchers regarded this phase as uninteresting.

The Tallying Phase

Taking a set of encrypted ballots and transforming it, in a universally verifiable manner, into a tally (together with a proof of correctness) is a nice cryptographic mathematical problem.

Researchers *really* liked this problem and spent decades developing and improving solutions.

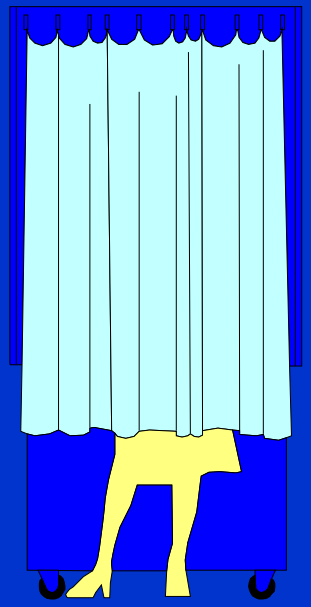
Fundamental Tallying Decision

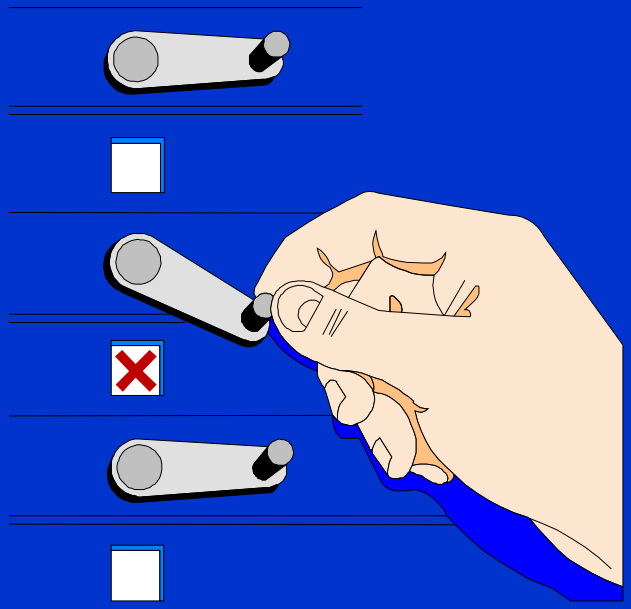
You have essentially two paradigms to choose from …

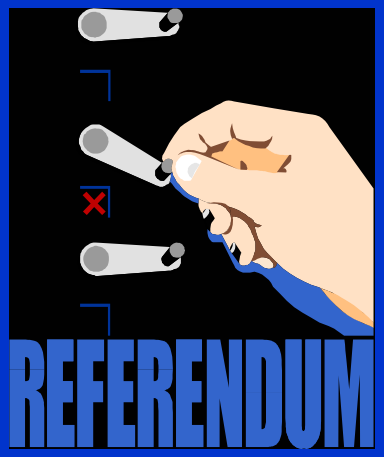
- Anonymized Ballots
- Ballotless Tallying

Anonymized Ballots

Ballotless Tallying







A Fundamental Trade-Off

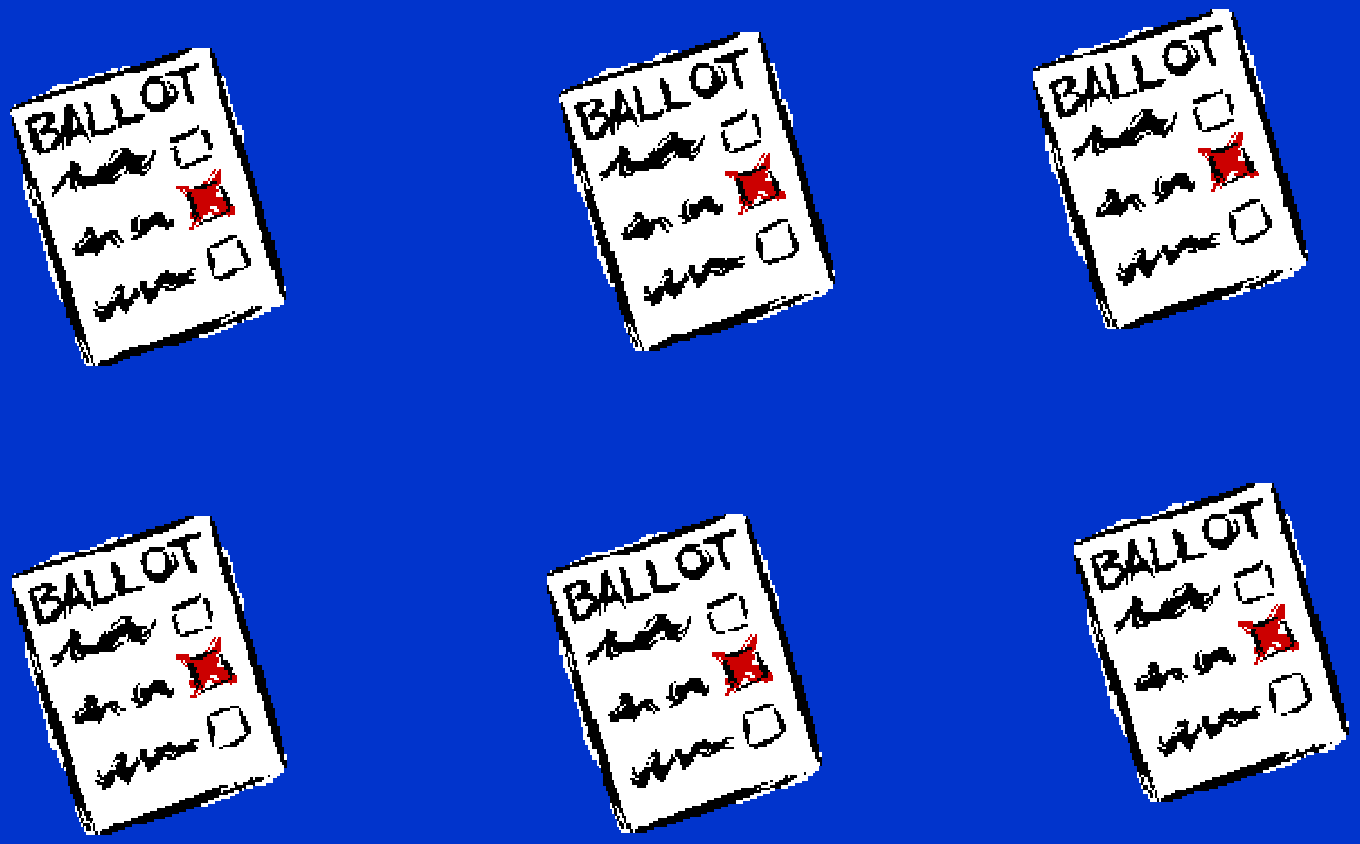
- Ballots simplify write-ins and other "non-standard" options.
- Non-standard options can compromise privacy.

Anonymized Ballots

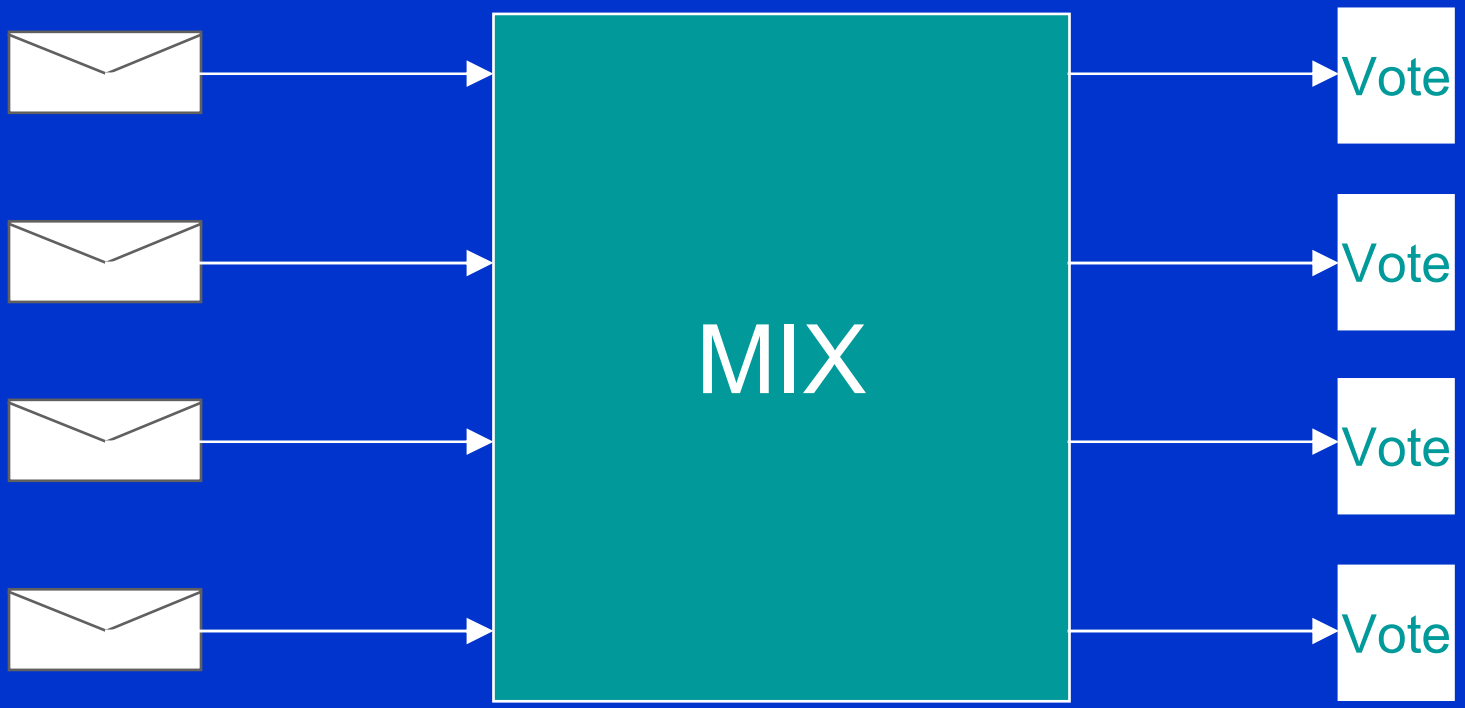
The Mix-Net Paradigm

Chaum (1981) …

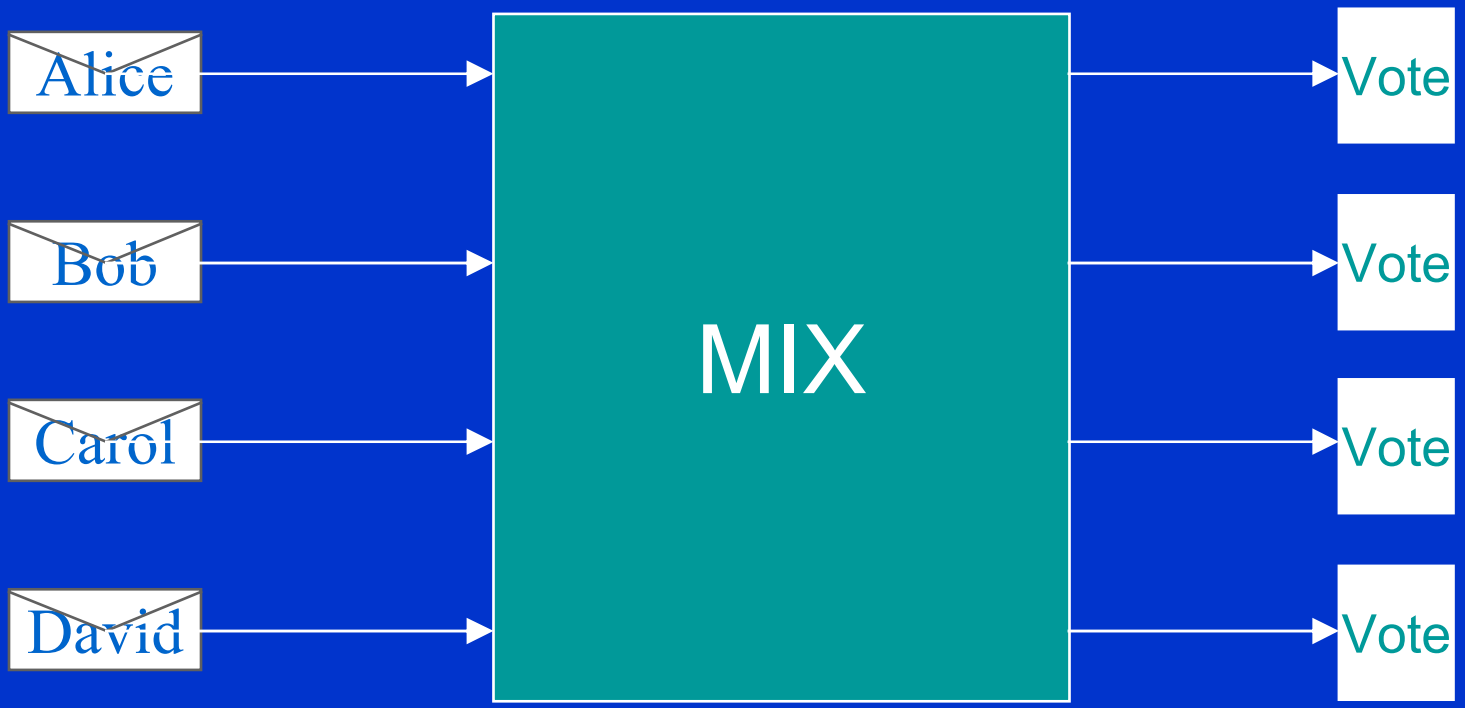
The Mix-Net Paradigm



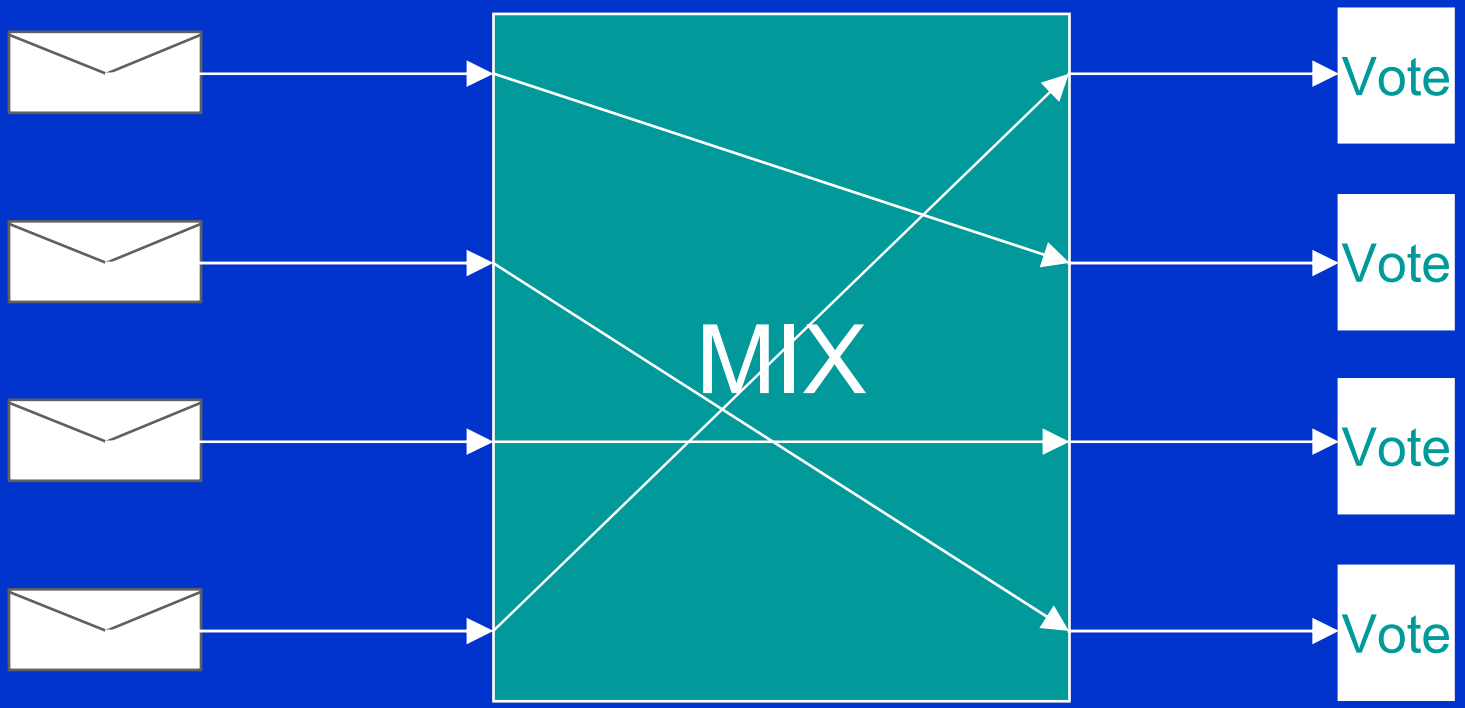
The Mix-Net Paradigm



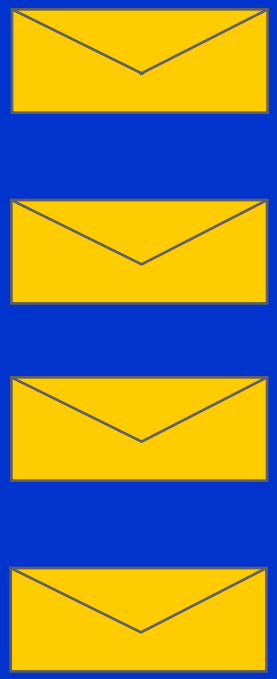
The Mix-Net Paradigm



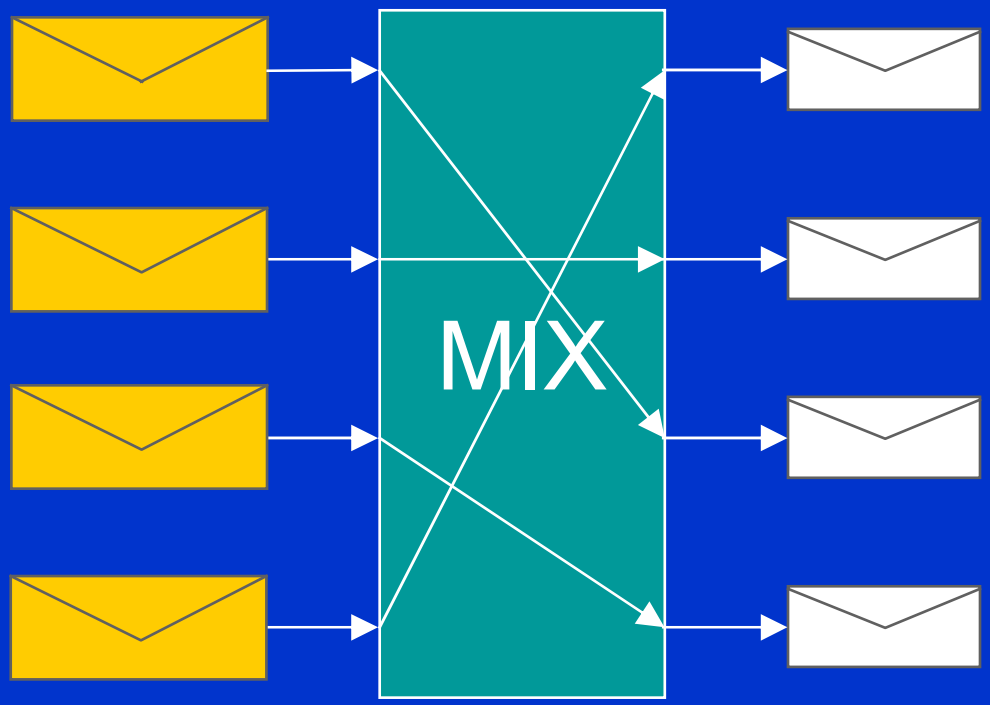
The Mix-Net Paradigm



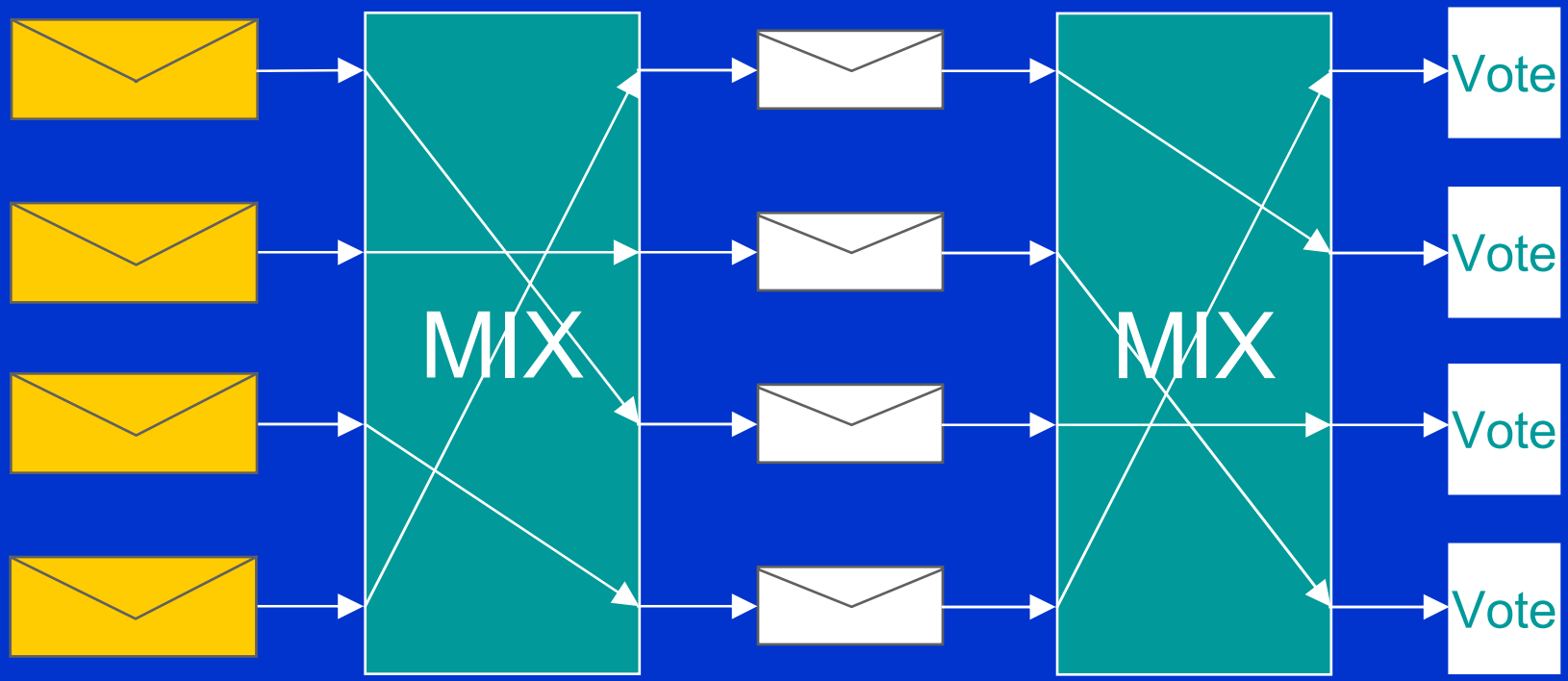
Multiple Mixes



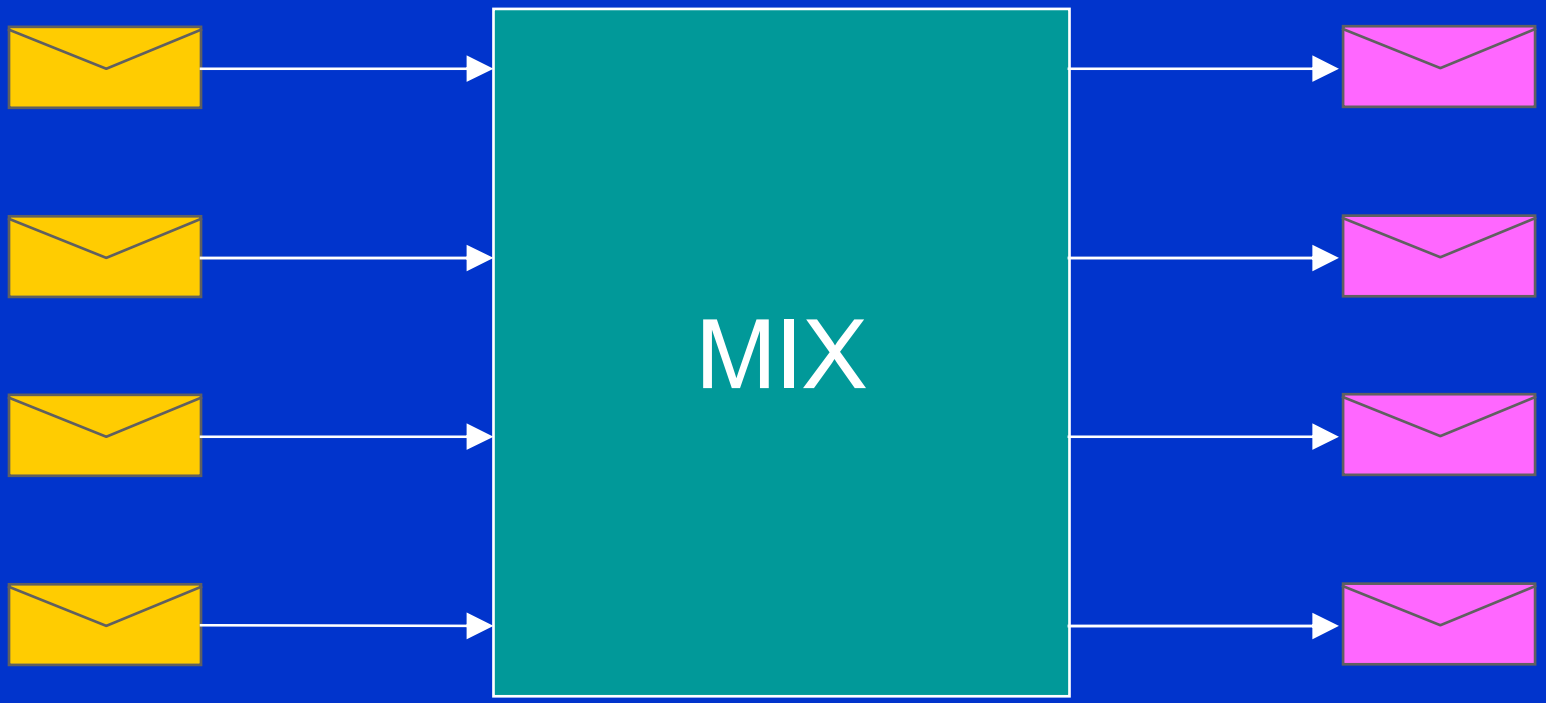
Multiple Mixes



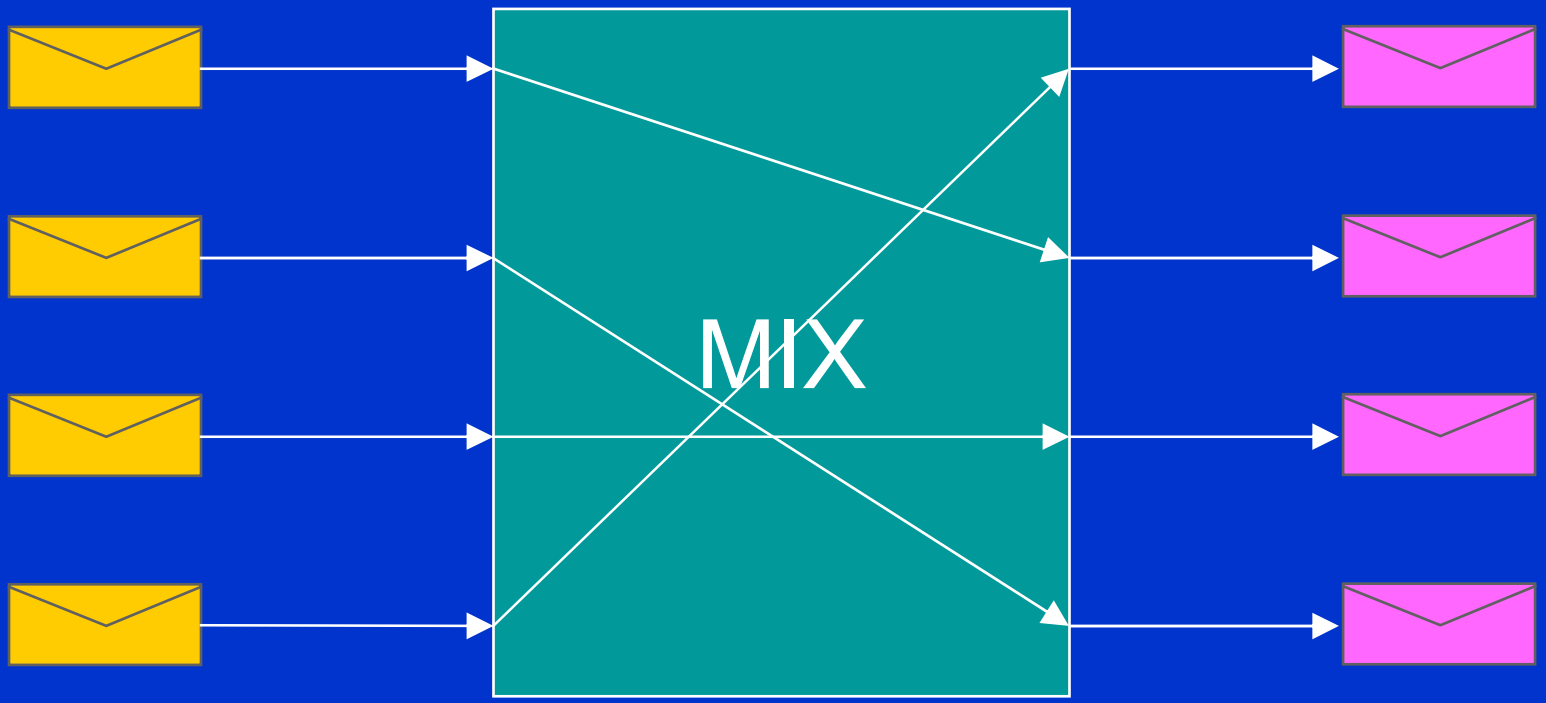
Multiple Mixes



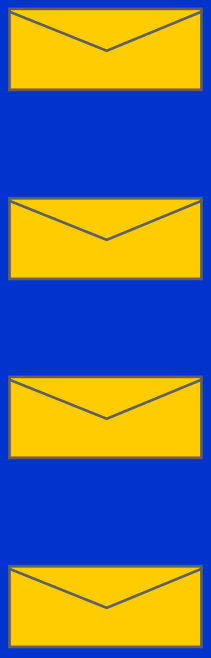
A Re-encryption Mix



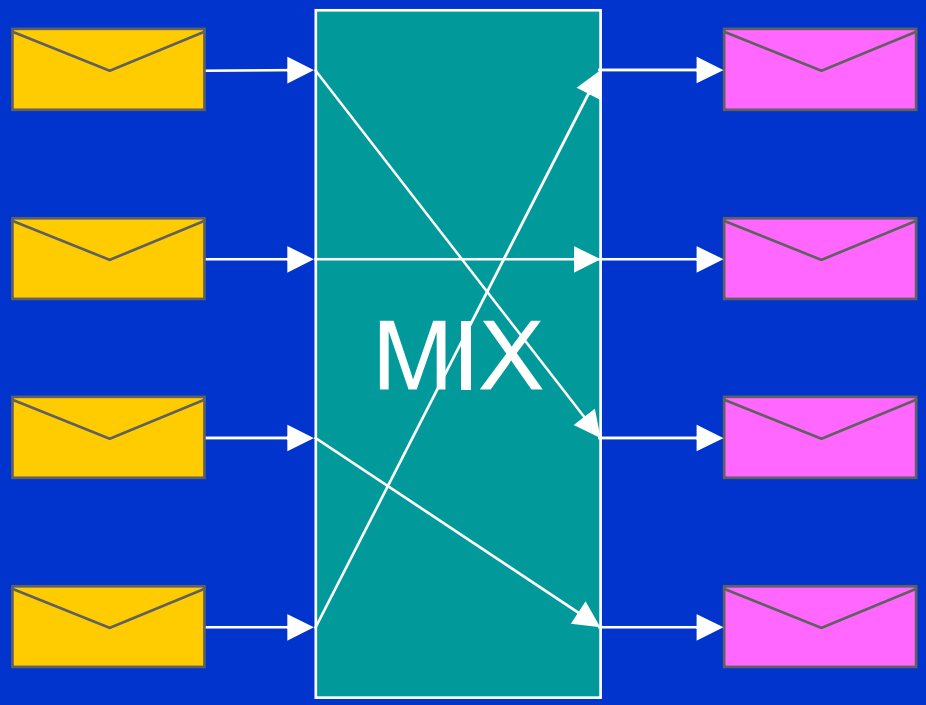
A Re-encryption Mix



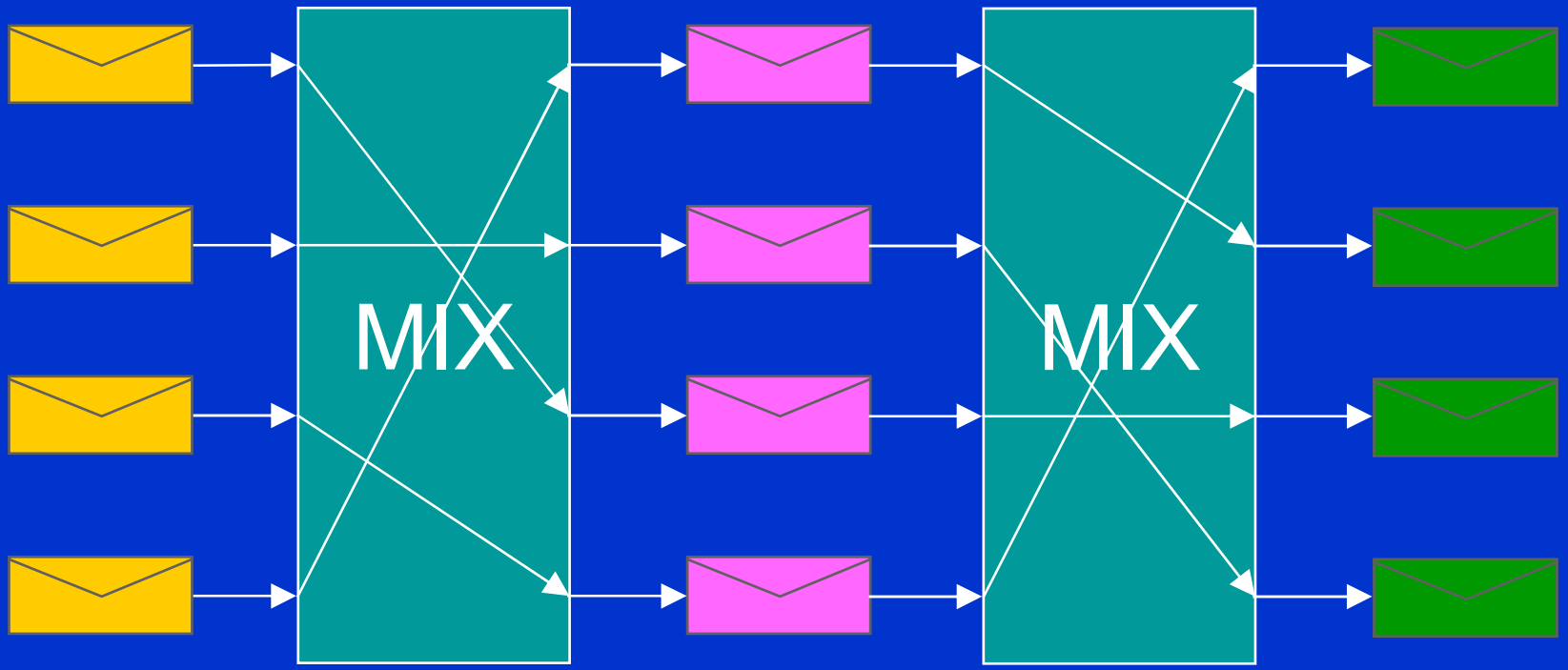
Multiple Re-encryption Mixes



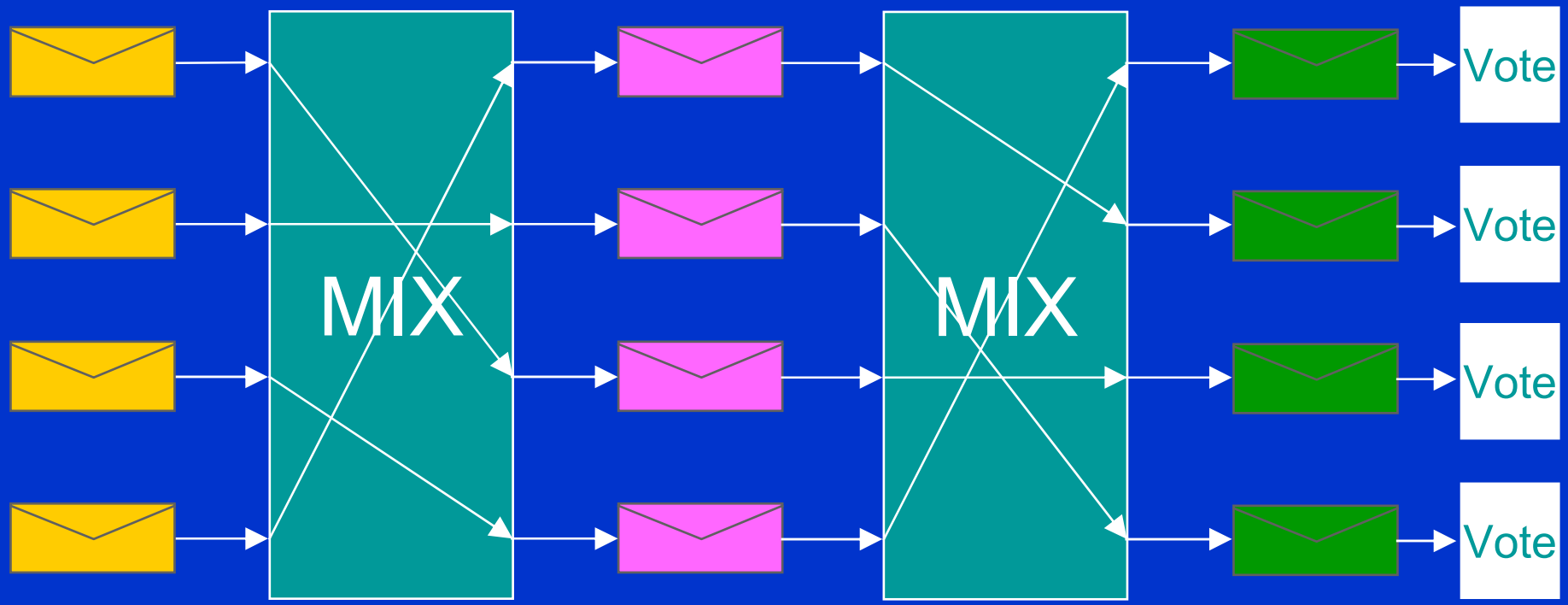
Multiple Re-encryption Mixes



Multiple Re-encryption Mixes



Multiple Re-encryption Mixes



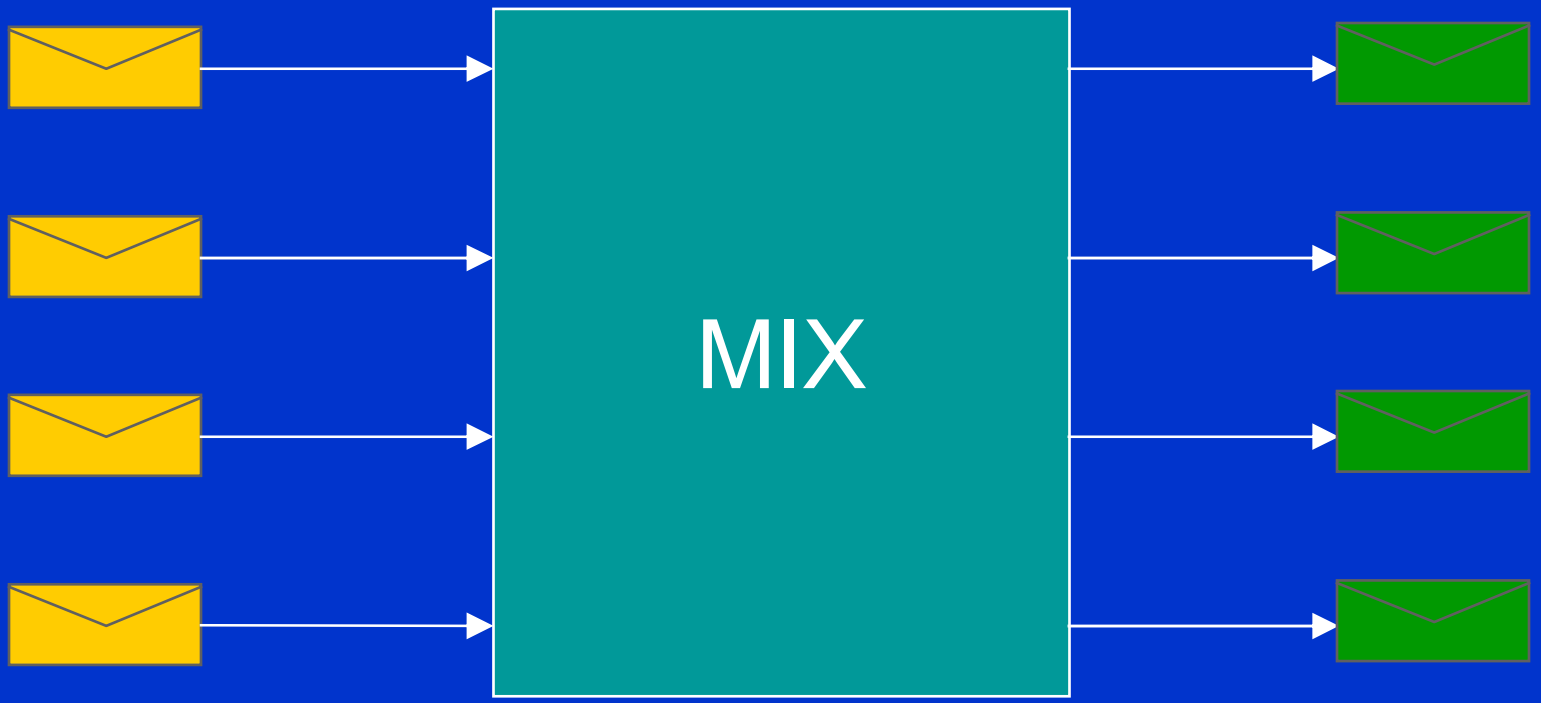
Verifiability

Each re-encryption mix provides a mathematical proof that it's output is a permutation of re-encryptions of its input.

Any observer can verify this proof.

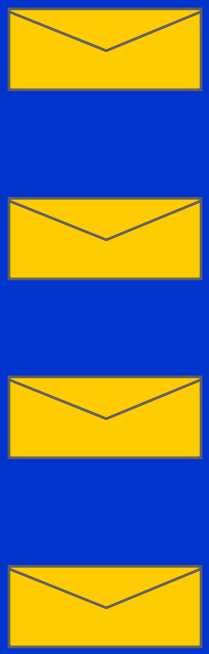
If a mix's proof is invalid, its mixing will be bypassed.

Constructing a Verifiable Mix



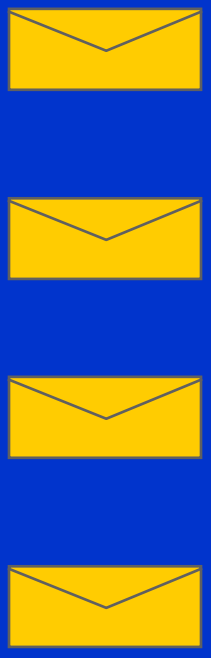
Constructing a Verifiable Mix

Initial Ballot Set

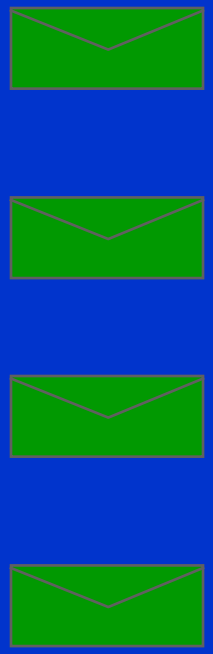


Constructing a Verifiable Mix

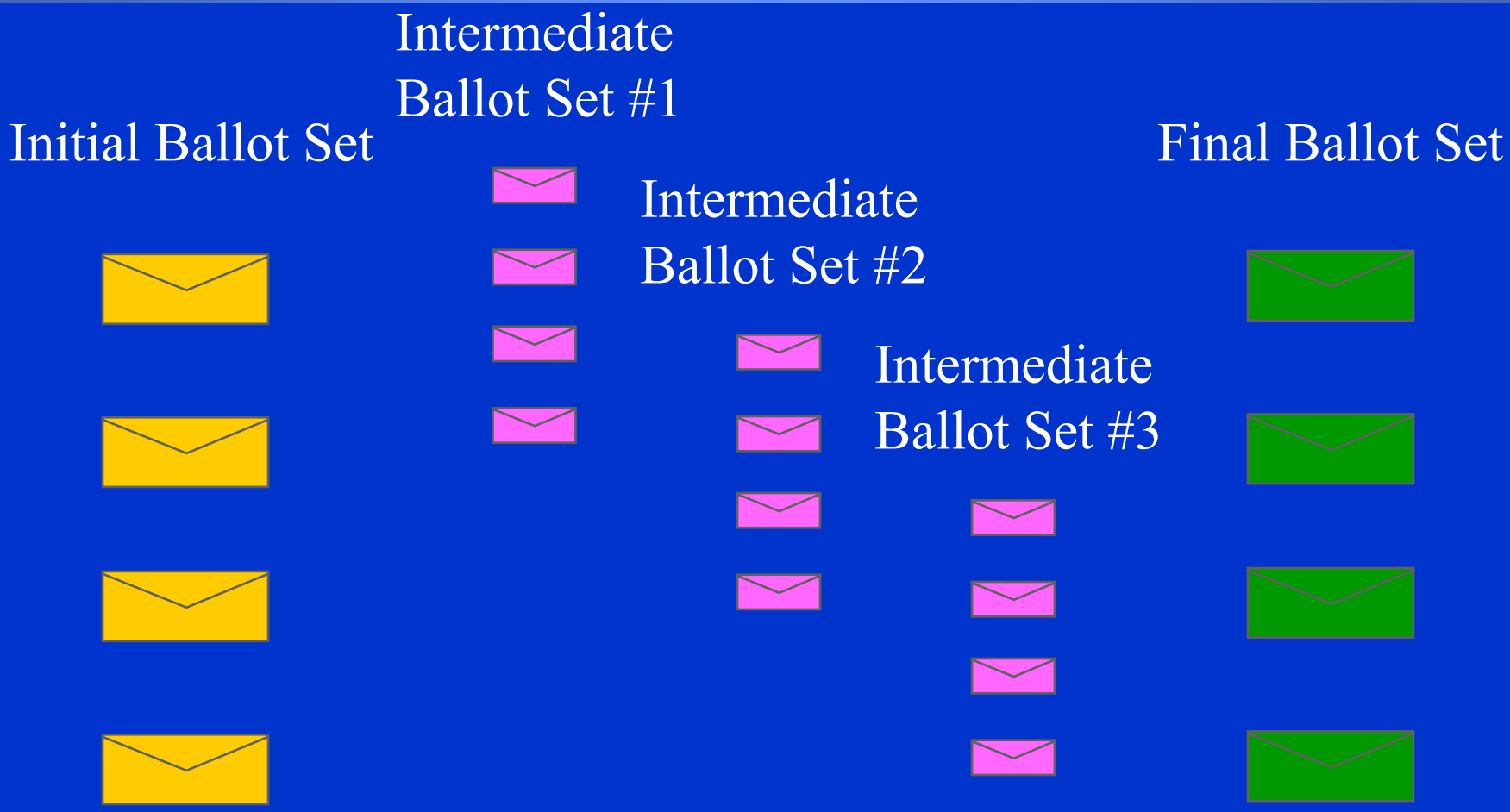
Initial Ballot Set



Final Ballot Set



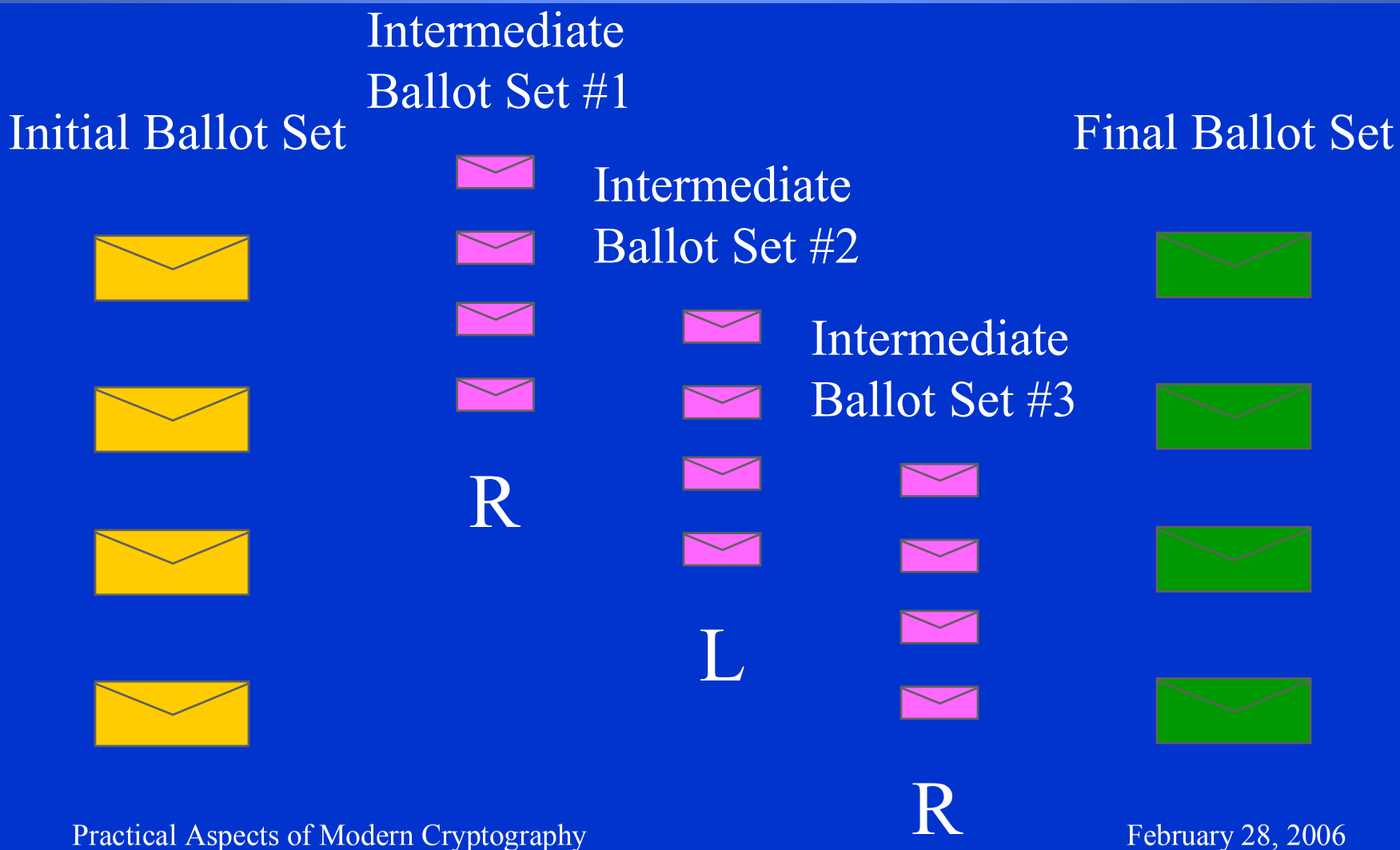
Constructing a Verifiable Mix



Constructing a Verifiable Mix

- Accept a one-bit challenge (L/R) for each intermediate ballot set.
- "Open" the indicated side of each intermediate ballot set to show that it is a valid (permuted) re-encryption of either the initial ballots or the final ballots.
- Generate these challenges non-interactively.

Constructing a Verifiable Mix



Unconditional Verifiability

- If the final ballot set *does not* match the initial ballot set, the chance of matching an intermediate ballot set to the indicated initial/final set is no more than 0.5.
- 100 intermediate ballot sets yields an error probability of no more than 2^{-100}.
- Challenge values may be collected non-interactively.

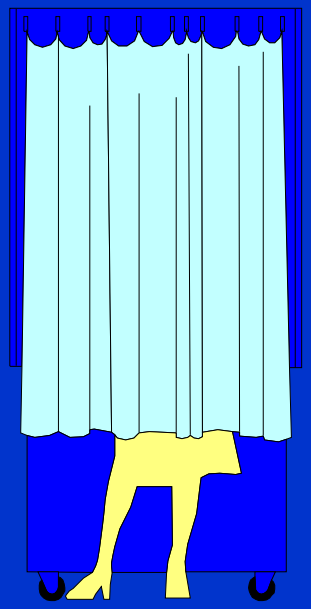
Mix-Net Properties

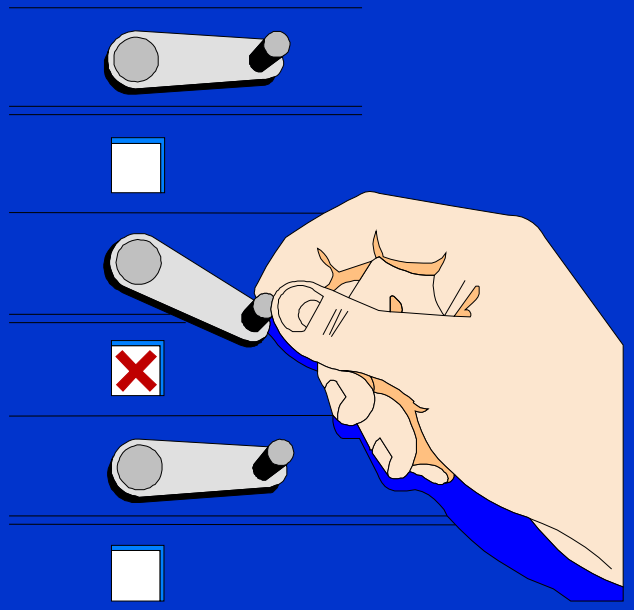
- The integrity of a mix-net is *not* dependent on any unproven assumptions – only the inability of a mix to predict the challenges it receives.
- Privacy in a mix-net *is* dependent upon the mixers and is no better than that provided by the encryption – a cryptographic break-through could compromise privacy.

Ballotless Tallying

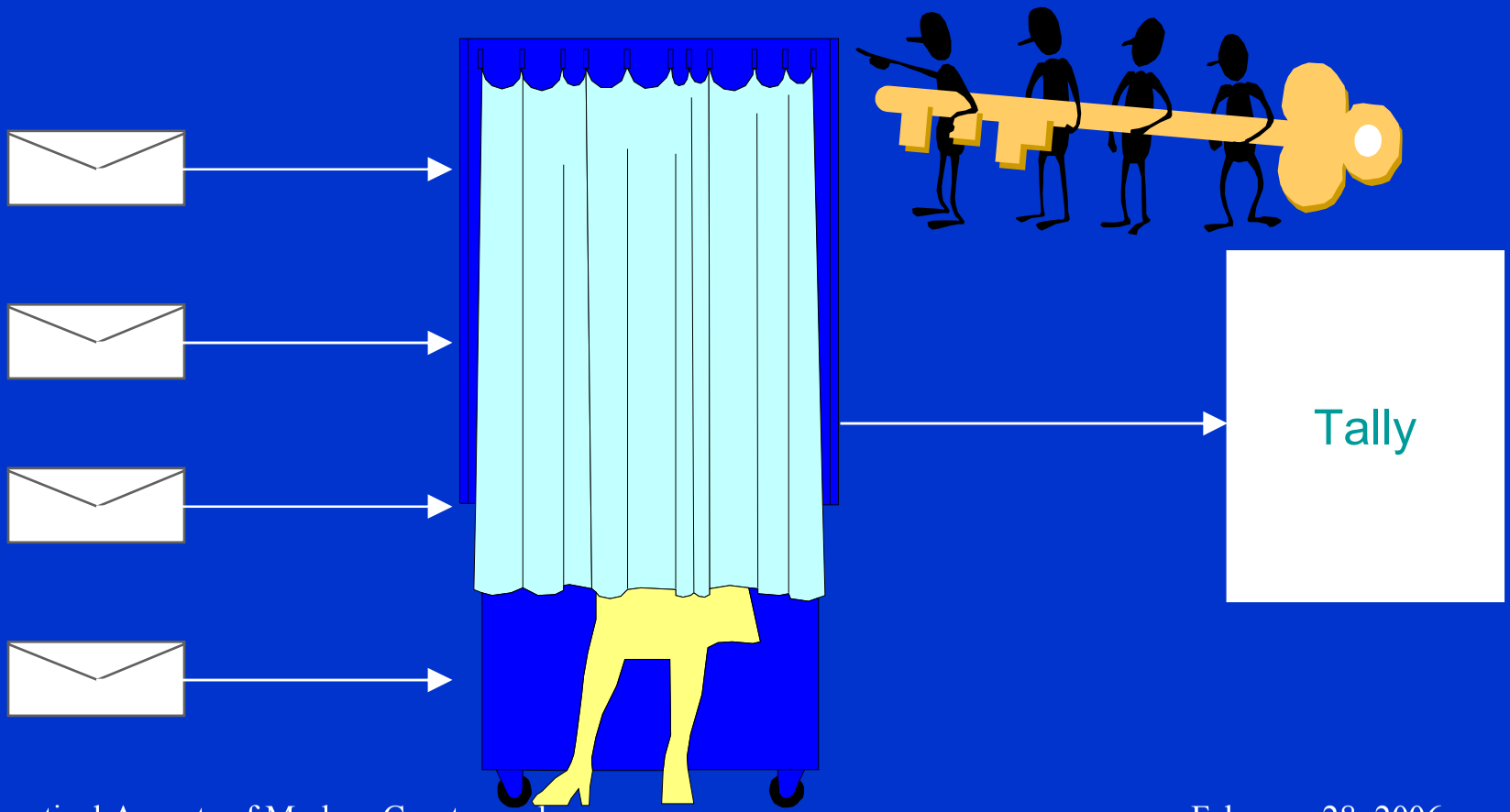
The Homomorphic Paradigm

Benaloh (Cohen), Fischer (1985) …

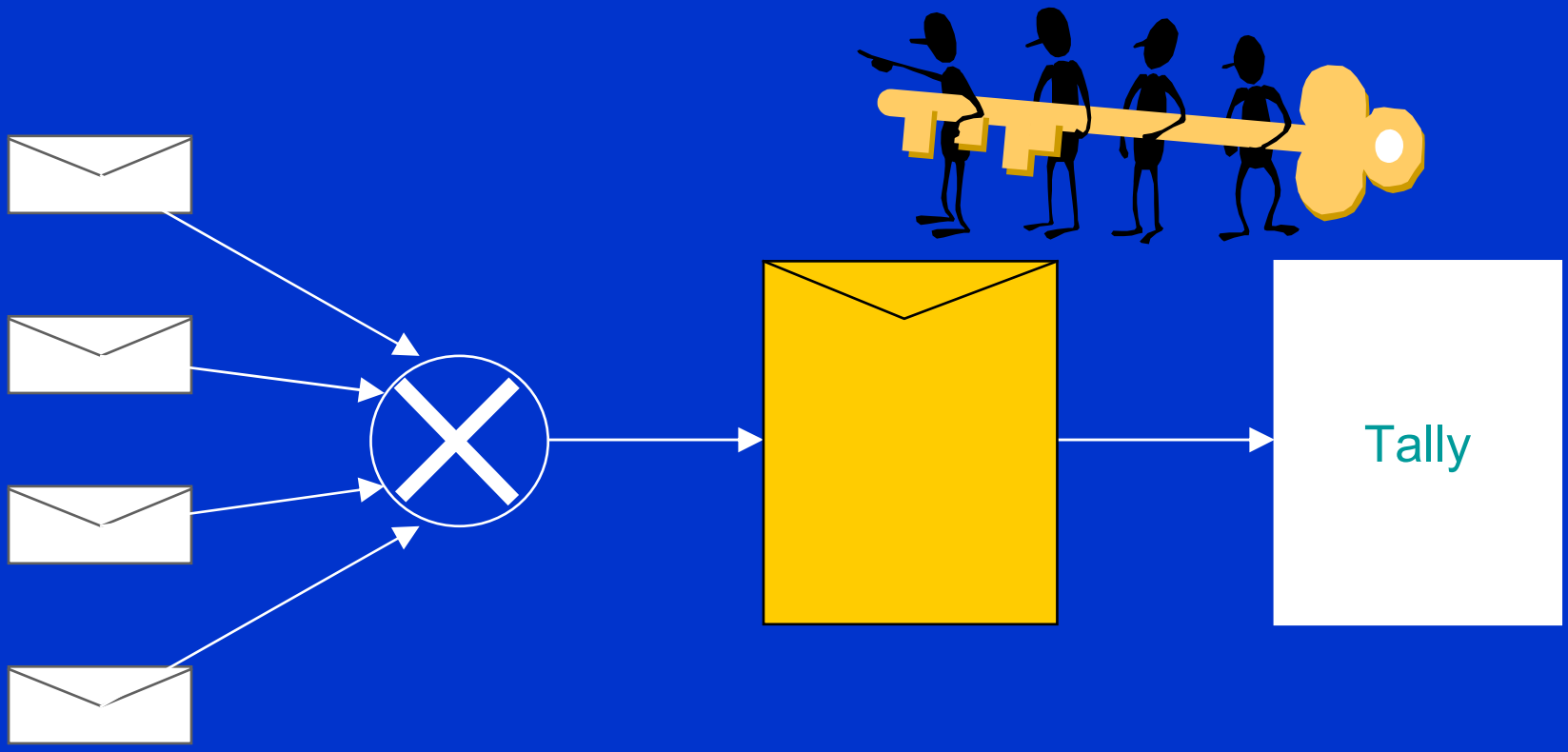




The Homomorphic Paradigm



The Homomorphic Paradigm



Homomorphic Techniques

Alice	0
Bob	0
Carol	1
David	0
Eve	1

Homomorphic Techniques

Alice	0
Bob	0
Carol	1
David	0
Eve	1
	$\Sigma =$

Homomorphic Techniques

Alice	0
Bob	0
Carol	1
David	0
Eve	1
$\Sigma =$	
2	

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
			$\Sigma =$	$\Sigma =$	$\Sigma =$

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
			$\Sigma =$	$\Sigma =$	$\Sigma =$
			3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
			$\Sigma =$	$\Sigma =$	$\Sigma =$
		$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
			$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
	$\Sigma =$		$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

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

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
	$\Sigma =$		$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
			$\otimes$ =	$\otimes$ =	$\otimes$ =

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
			$\otimes =$	$\otimes =$	$\otimes =$
			3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
			$\otimes =$	$\otimes =$	$\otimes =$
			3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
			$\otimes =$	$\otimes =$	$\otimes =$
		$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
			$\otimes =$	$\otimes =$	$\otimes =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0		3	-5	2
Bob	0		-4	5	-1
Carol	1		2	-3	2
David	0		-2	-1	3
Eve	1		4	-1	-2
	$\Sigma =$		$\otimes =$	$\otimes =$	$\otimes =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

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

The *product* of the *encryptions* of the votes constitutes an *encryption* of the *sum* of the votes.

Homomorphic Techniques

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.

Homomorphic Techniques

Robustness can be added by using a threshold scheme instead of a simple sum to share each vote.

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	$= \Sigma$	3	-5	2
Bob	0	$= \Sigma$	-4	5	-1
Carol	1	$= \Sigma$	2	-3	2
David	0	$= \Sigma$	-2	-1	3
Eve	1	$= \Sigma$	4	-1	-2
	$\Sigma =$		$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	$= \Sigma$	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	3	-5	2
Bob	0	= P(0)	-4	5	-1
Carol	1	= P(0)	2	-3	2
David	0	= P(0)	-2	-1	3
Eve	1	= P(0)	4	-1	-2
	$\Sigma =$		$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	= P(0)	3	-5	4

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1
	$\Sigma =$		$\Sigma =$	$\Sigma =$	$\Sigma =$
	2	= P(0)	4	6	8

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1
			$\otimes$ =	$\otimes$ =	$\otimes$ =
			4	6	8

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1
			⊗ =	⊗ =	⊗ =
			4	6	8

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1
			$\otimes$ =	$\otimes$ =	$\otimes$ =
	2	= P(0)	4	6	8

Homomorphic Techniques

			X_1	X_2	X_3
Alice	0	= P(0)	2	4	6
Bob	0	= P(0)	1	2	3
Carol	1	= P(0)	2	3	4
David	0	= P(0)	-2	-4	-6
Eve	1	= P(0)	1	1	1
	$\Sigma =$		$\otimes =$	$\otimes =$	$\otimes =$
	2	= P(0)	4	6	8

The Encryption Phase

How can voters turn their intentions into encrypted ballots?

Any device that can perform this task could have vulnerabilities, intentional back doors, be subject to viruses, etc.

The Encryption Phase

Requirements of ballot encryption devices

- Must accurately encrypt voter intentions
- Need *not* know voter identities
- Need *not* authenticate voters right to vote
- Need *not* limit people to a single use
- Need *not* cast votes

Unstructured Auditing

- Anyone … voter/inspector/observer is free to create votes at any time during an election.
- Any "uncast" votes can be opened at any time – either by the device itself or by adding the vote to a separate "test vote" queue.

A Simple Audit

- Go into vote encryption booth.
- Create 4 encrypted ballots: 2 for each of candidate A and candidate B.
- Leave vote encryption booth with 4 encrypted ballots.
- Take one of the encrypted votes for each of A and B and have them decrypted.
- Cast one of the 2 remaining encrypted votes.

Auditing

Note that it's *not* necessary for all voters to audit vote encryption devices – a tiny fraction of voters and/or election inspectors would suffice, and there are more user-friendly ways to effect auditing.

E.g. 100 auditing events would probably detect a 1% fraud rate.

A Fundamental Limitation

Whenever a ballot is created for the voter, there seems to be no way to distinguish between a vote-creation device attempting to cheat and a voter claiming that a properly functioning device attempted to cheat the voter.

In Practice?

Voter

- Go to your polling station and (possibly without even signing in) go to a voting station.
- Enter your preferences and at the end press (perhaps) one of two buttons.
- Receive a machine *and* human readable card.
- Go to a poll worker, sign-in, and run your card through a reader.
- Take the card home and (if you wish) use it to verify your vote on-line.

In Practice?

Back End

- Receive all votes and post them on-line (perhaps even together with voter names).
- Allow anyone to (sequentially) scramble (mix) the votes and provide a proof of correct mixing. Post all such mixings and proofs on-line.
- Have the final mixed ballots decrypted together with proof of correct decryption. Post the decryptions together with their proofs.

In Practice?

Observers

- Use voting stations to create ballots anytime desired. Test ballots for validity.
- Verify posted ballot mixing and decryptions using posted proofs.

Properties

- Cryptographically verified election technologies can achieve universal end-to-end verifiabilty, while paper can only provide limited voter verifiability.
- This is a substantially different paradigm that emphasizes validations of *elections* rather than *election equipment*.
- A cryptographic election can be verified externally without ever having to inspect the system hardware or software.

Scorecard

	Crypto Based	Paper Based

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability		

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility		

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely
Robustness		

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely
Robustness	Wholesale failure is possible	

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely
Robustness	Wholesale failure is possible	Only retail failure is possible

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely
Robustness	Wholesale failure is possible	Only retail failure is possible
Overall		

Scorecard

	Crypto Based	Paper Based
Accuracy/ Verifiability	Full end-to-end by anyone	Voter only as far as ballot box
Privacy/ Coercibility	Cannot be proven absolutely	Cannot be proven absolutely
Robustness	Wholesale failure is possible	Only retail failure is possible
Overall	?	?

Conclusions

Conclusions

- Keep an open mind.

Conclusions

- Keep an open mind.
- Think critically.

Conclusions

- Keep an open mind.
- Think critically.
- Vote!