Computer Science 220r Notes
Harvard Fall 2011

Prof. Michael Rabin
Compiled by Osbert Bastani
Edited by Anna Gommerstadt

December 9, 2011

1 Thursday, 9/1

Consider the situation where Alice wants to send Bob a message across an insecure channel.

Definition: In **private key cryptosystem**, Alice and Bob share a key k. Alice sends encrypted message $C = \text{Enc}(M, k)$, and Bob decrypts $M = \text{Dec}(C, k)$.

Unfortunately, establishing a key is difficult. Also, every pair of people must share a key.

Definition: In a **public key cryptosystem**, Bob has a private key d_B and a public key e_B. Alice sends encrypted message $C = \text{Enc}(M, e_B)$ and Bob decrypts $M = \text{Dec}(C, d_B)$.

Definition: In a **digital signature**, Bob has contract C for Alice that he wishes to sign. Bob has a private sign-key d_B. He computes $s = \text{SIGN}(C, d_B)$, so that (C, s) is signed.

Definition: A **one way function** is a function $f : \{0,1\}^n \to \{0,1\}^m$, where $f(x)$ is easy to compute, but given $y = f(x)$, it is hard to find ANY x' such that $f(x') = y$.

The following can be accomplished by using a one way function.

Definition: In a **commitment**, Bob has data d, say a bid on an Auction. He wants to send this to Alice, say the auctioneer, without revealing the value until after some time T, so he sends Alice $C = \text{COM}(d)$. Alice wants to guarantee that Bob does not change his bid. v **Definition:** In **multiparty computations**, $P_1, ..., P_n$ hold data $x_1, ..., x_{10}$, and they want to perform computations on this data without revealing the data.

This has the following application.

Definition: In **time-lapse cryptography**, we have ciphertexts $e_1, ..., e_n$ encrypted by key d_i. At time T_i, key d_i should be revealed, but not before. Also, no client (or small group of clients) can stop the key from being revealed.

Definition: In a **zero-knowledge proof**, P claims to know a fact, e.g. factorization of a large integer $n = pq$. Then P can prove this to a verifier V, who gives P random challenges, without letting V know the fact.

This has the following applications.

Definiton: In **authentication**, $n_B = p_B q_B$ is public. If B wants to log into a server, he must prove that he knows the factorization to the verifier V, i.e. the server.

2 Tuesday, 9/6

Definition: In the **one time pad** Alice and Bob share $b_1...b_N$ chosen uniformly randomly in $\{0, 1\}$. To send message $m = m_1...m_N$, Alice sends $\text{Enc}(m) = m \oplus b$ and Bob decodes the message he receives by $m = \text{Dec}(c) = c \oplus b = m \oplus b \oplus b = m$.

Theorem: The one time pad is provably unbreakable.

2.1 Number Theory

Proposition: If $2^k - 1$ is prime, then k is prime.

Theorem: For any $n \in \mathbb{Z}$, n admits a unique factorization into primes

$$n = p_1^{e_1}...p_k^{e_k}.$$

Theorem: There are infinitely many prime numbers.

Proof: Assume to the contrary that $q_1, ..., q_k$ enumerate the primes. Then for all $n \geq 1$, we have

$$n = q_1^{e_1}...q_k^{e_k},$$

where $0 \leq e_i \leq \log_2 n$, so

$$n = \#\{x : x \leq n\} \leq (\log_2 n + 1)^k,$$

but this is absurd for n sufficiently large.

Definition: Define the prime counting function by

$$\pi(x) = \#\{p : p \leq x\}.$$

Theorem: (Chebychev) If $100 < x$, then

$$\frac{0.9x}{\ln x} \leq \pi(x) \leq \frac{1.1x}{\ln x}.$$

Theorem: (prime number theorem) We have

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1.$$

Theorem: Let $n \in \mathbb{N}$. There exists a randomized algorithm to test whether n is prime that runs in time $O(\log n)$ with error at most $\frac{3}{4}$.

Algorithm: We want to construct $n = pq$, where n has N digits. Set $x = 2^N$, so $2x = 2^{N+1}$. We want to choose $x \leq p, q < 2x$. To do so, randomly test $x \leq p' < 2x$ and test it for primality. This algorithm is efficient by the prime number theorem, since

$$\#\{x \leq p \leq 2x\} \approx \frac{x}{\ln x},$$

so the probability of success is

$$\frac{1}{\ln x}.$$

Definition: The group $\mathbb{Z}/n\mathbb{Z}$ is the cyclic group of order n.

Algorithm: To compute $a^b \bmod c$, write

$$b = \epsilon_{n-1}...\epsilon_0,$$

and let L_i, $1 \leq i \leq m$ be the set such that $\epsilon_{L_i} = 1$. Then compute

$$a^b \bmod c = \left(a^{2^{L_m}} \bmod c\right) ... \left(a^{2^{L_0}} \bmod c\right) \bmod c.$$

3 Thursday, 9/8

3.1 Group Theory

Definition: A group G is a structure

$$G = (G, \cdot, e),$$

where

1. G is a set,
2. $\cdot$ is an operation such that for $x, y \in G$, $x \cdot y \in G$, i.e. $\cdot G \times G \to G$,
3. $\cdot$ is associative, i.e. $(x \cdot y) \cdot z = x \cdot (y \cdot z)$,
4. $e \in G$ is the unit in G, i.e. $e \cdot x = x \cdot e = x$,

5. $\forall x \in G$, $\exists y \in G$ such that $x \cdot y = y \cdot x = e$, and write $y = x^{-1}$.

Example: Let

$$\mathbb{Q} = \left\{ r = \frac{s}{t} : s, t \in N - \{0\} \right\}.$$

Then $G = (\mathbb{Q}, \cdot, 1)$ is a group.

Example: $G = (\mathbb{Z}, +, 0)$ is a group.

Example: Let

$$\mathcal{M}_{2,2} = \left\{ M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} : |M| \neq 0 \right\},$$

and let

$$I_{2,2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then $G = (\mathcal{M}_{2,2}, \cdot, I_{2,2})$ is a group.

Definition: If $k \in \mathbb{N}$, and $g \in G$, then $g^k = g \cdot ... \cdot g$ (k times), and $g^0 = e$.

Theorem: Let G be finite, $|G| = n$, and $g \in G$. Then $\exists 1 \leq k \leq n$ such that $g^k = e$.

Proof: If $g = e$, then take $k = 1$. Otherwise, consider

$$\{g, g^2, ..., g^n\} \subset G.$$

Since there are n elements in G, two of these must be equal, say $g^s = g^t$ with $t > s$. Then

$$e = g^s g^{-s} = g^t g^{-s} = g^{t-s},$$

so take $k = t - s$.

Definition: The smallest $k \geq 1$ for which $g^k = e$ is called the **order** of $g \in G$, and denote it by $\text{ord}(g) k =$.

Theorem: If $|G| = n$, $g \in G$, $\text{ord}(g) = m$, then $m | n$.

Example: Consider groups G where $|G| = p$, where p is prime. Let $e \neq g \in G$. Then $\text{ord}(g) = k > 1$. Since $k | p$, we must have $k = p$. Therefore $G \cong \mathbb{Z}/p\mathbb{Z}$ is a cyclic group of order p, and every $e \neq g \in G$ generates G. Also,

$$(\mathbb{Z}/p\mathbb{Z})^* = \{1, ..., p - 1\}$$

is a multiplicative group of order $p - 1$.

Definition: Let G such that $|G| = p$. Let $g, h \in G$. Then there exists $m \in G$ such that $g^m = h$. Denote the smallest such m by $m = \log_g h$. The **discrete logarithm problem** is to compute $m = \log_g h$.

3.2 El Gamal

Assumption 1: There exists a class of groups G such that $|G| = p$ where the discrete logarithm problem is intractable, i.e. there is no polynomial time algorithm to compute $\log_g h$.

Assumption 2: (Diffie-Hellman) Given g, g^a, and g^b, it is difficult to compute g^{ab}.

Algorithm: (El Gamal) Alice want to communicate using AES.

1. Public: G, with $|G| = p$, and $e \neq g \in G$.
2. Alice chooses $a \in [e, p-1]$ uniformly randomly.
3. Bob chooses $b \in [0, p-1]$ uniformly randomly.
4. Alice computes $h_a = g^a$ and sends this to Bob.
5. Bob computes $h_b = g^b$ and sends this to Alice.
6. Alice computes $h_b^a = (g^b)^a = g^{ab}$.
7. Bob computes $h_a^b = (g^a)^b = g^{ab}$.

Now Alice and Bob share g^{ab}.

Proposition: This is secure if Assumption 2 holds.

Attack: (Man-in-the middle) If Eve intercepts g^a and g^b, she can replace them with $g^{a'}$ and $g^{b'}$ respectively, where a' and b' are known to her. It is not possible for Alice or Bob to determine that this has occured, and now Eve can translate between them.

Algorithm: (El Gamal public key) Now consider a group G with $e \neq g \in G$. Alice has a message $M \in G$ for Bob.

1. Bob chooses $a \in [0, p-1]$ uniformly randomly.
2. Bob sets $h_B = g^a$ to be his public key and $a_B = a$ to be his private key.
3. Alice chooses $b \in [0, p-1]$ uniformly randomly.
4. Alice computes
$$(x, y) = \text{Enc}(M, h, b) = (g^b, h^b M),$$
and sends these to Bob.
5. Bob computes
$$\text{Dec}(x, y, a) = x^{-a} y = g^{ab} h^b M = h^{-b} h^b M = M.$$

Proposition: Given Assumption 2, this is secure.

Proof: The adversary sees $h^b M$. If he can compute M, then he would be able to compute

$$\frac{h^b M}{M} = h^b = g^{ab},$$

contradicting Assumption 2.

Definition: A cipher is **semantically secure** if given $C = E(M)$, where it is known that $M \in \{M_1, M_2\}$, it is still impossible to determine whether $M = M_1$ or $M = M_2$.

Theorem: RSA is not semantically secure.

4 Tuesday, September 13

Definition: Given $a, b \in \mathbb{Z}$, define their **greatest common divisor** to be

$$\gcd(a, b) = (a, b) = \sup\{d \in \mathbb{Z} : d|a, d|b\}.$$

Proposition: If $d'|a$ and $d'|b$, then $d'|d$.

Algorithm: To compute the greatest common divisor d of a and b,

1. find the prime factorizations
$$a = p_1^{e_1} ... p_k^{e_k}; \qquad b = p_1^{f_1} ... p_k^{f_k},$$
2. return
$$d = p_1^{\min\{e_1, f_1\}} ... p_k^{\min\{e_k, f_k\}}.$$

However, integer factorization is assumed to be hard. The following algorithm is polynomial time in $\log n$.

Theorem: If $a, b \in \mathbb{N}$, $a > b$, then $(a, b) = (a - b, b)$.

Proof: If $d \mid a$ and $d \mid b$, then $d \mid (a - b)$, i.e.

$$\{d : d \mid a, d \mid b\} = \{d : d \mid a, d \mid (a - b)\},$$

so their supremum are equal.

Proof: If $d' \mid a$ and $d' \mid (a-b)$, then $d' \mid (a-b+b) = a$ so $d' \mid d$. Conversely, if $d \mid a$ and $d \mid b$, then $d \mid (a-b)$ so $d \mid d'$. Hence $d = d'$.

Algorithm: To compute the greatest common divisor of a and b,

1. write $a = 2^{k_1} a'$ and $b = 2^{k_2} b'$, $2 \nmid a'$ and $2 \nmid b'$,

2. let $k = \min\{k_1, k_2\}$,

3. since $2^k \mid d$, we can write $d = 2^k d'$, where $2 \nmid d'$,

4. if $a' = b'$ return a'

5. otherwise assume $a' > b'$ and return $\gcd(a' - b', b')$.

Proof: The algorithm terminates by infinite descent since $a' - b'$ is even. In particular, note that a' is even, so if a is even then $a' < \frac{a}{2}$. In each recursive call to the algorithm, a is even, so the algorithm runs in time $O(\log_2 n)$. Correctness follows from the previous theorem.

Theorem: We have $(a, b) = d \Rightarrow \exists x, y \in \mathbb{Z}$ such that $ax + by = d$.

Proof: Exercise.

Definition: We say that a and b are relatively prime if $(a, b) = 1$.

Theorem: We have $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$ is a commutative group under multiplication.

Proof: If $a, b \in \mathbb{Z}_p^*$, then $ab \bmod p \in \mathbb{Z}_p^*$, since $a, b < p$ so they have no common factors with p, so the set is closed under multiplication. It is clear that the group is associative and has unit. To show inverses, note that $(a, p) = 1$, so there exists $x, y \in \mathbb{Z}$ such that $ax + bp = 1$, so $ax = 1 \bmod p$, so $x = a^{-1} \bmod p$. This group is also commutative, since multiplication is commutative.

Theorem: (Fermat's Little Theorem) Let G be a finite commutative group, with $|G| = n$. Then if $g \in G$, we have $g^n = e$.

Proof: Let $G = \{e, g_1, ..., g_{n-1}\} = \{eg, g_1 g, ..., g_{n-1} g\}$, since $g' \neq g'' \Rightarrow g'g \neq g''g$. Therefore

$$eg_1 ... g_{n-1} = (eg)(g_1 g)...(g_{n-1} g) \Rightarrow g^n = e.$$

Definition: A prime q is a **Sophie Germaine prime** if $q = 2p + 1$ where p is prime.

It is not even known whether there are infinitely many Sophie Germaine primes.

Conjecture: The density of primes is asymptotic to $\frac{x}{\ln x}$. Hence

$$\#\{p \leq x : p \text{ is Sophie Germaine}\} \sim \frac{x/2}{(\ln x)^2}.$$

Algorithm: To find a Sophie Germaine prime,

1. choose $n \in [2^N, 2^{N+1}]$,

2. if n is prime, test $2p + 1$.

Proposition: This algorithm is efficient if the above conjecture holds.

Definition: If p is prime, and $y \in \mathbb{Z}_p^*$, then y is a **quadratic residue** modulo p if $\exists x \in \mathbb{Z}_p^*$ such that $x^2 = y \bmod p$. We write

$$\left(\frac{y}{p}\right) = \begin{cases} 1 \text{ if } y \text{ is a quadratic residue} \\ -1 \text{ if } y \text{ is not a quadratic residue} \end{cases}.$$

Theorem: Half the elements of $\mathbb{Z}_p^*$ are quadratic residues.

Proof: This follows because $x^2 = (p-x)^2 \bmod p$. Since $\mathbb{Z}_p$ is a field, $x^2 = a \bmod p$ has at most two solutions, so half of the elements of $\mathbb{Z}_p^*$ are quadratic residues.

Definition: Let $q = 2p + 1$ be a Sophie Germaine prime. In $\mathbb{Z}_q^*$, the set of quadratic residues modulo q is a subgroup. Call it R_q. Then $|R_q| = \frac{q-1}{2} = p$, so this is a cyclic group of order p. We call this group the **Sophie Germaine group**, and also denote it by G_p.

Assumption: Let $q = 2p + 1$ for a large prime q. The discrete logarithm and Diffie-Hellman problem are intractable for R_q.

Recall: In the **Commitment problem**, Alice has a value $x \in [0, ..., p-1]$. She wants to send Bob or publish $\text{COM}(x) = C$ such that

1. at a later time, she reveals x and Bob can verify $\text{COM}(x) = C$,
2. $\text{COM}(x)$ reveals no information about x,
3. it is intractable for anyone to compute from C an $x_1 \neq x$ such that $\text{COM}(x_1) = \text{COM}(x)$, i.e. the commitment is computationally binding.

Algorithm: Let G_p be the Sophie Germaine group, and let $e \neq g_1, g_2 \in G_p$ be randomly chosen. Therefore $\log_{g_1} g_2$ is intractable. To commit an element $x \in G_p$,

1. Alice randomly chooses $r \in \mathbb{Z}_p$,
2. The commitment is $\text{COM}(r, x) = g_1^r g_2^x = C_{\text{Alice}}$,
3. To decommit, Alice sends Bob r and x.

Theorem: Given the assumption, this is a valid commitment algorithm.

5 Thursday, September 15

5.1 Commitment

Say Alice has a secret x she wants to commit to. Alice sends $C = \text{COM}(r, x)$ to Bob. After time T, Alice sends x and r to Bob, which he can check. We want

1. the item x to be information-theoretically secure without r,
2. the value of x is computationally binding on Alice (and others).

Algorithm: Let G_p be the multiplicative group of order p of quadratic residues modulo $q = 2p + 1$, where q is a Sophie Germaine prime. Every $1 \neq g \in G_p$ is a generator of $G_p = \{1, g, ..., g^{p-1}\}$. Let G_p, and $1 \neq g_1, g_2 \in G_p$ be public. Then $\log_{g_1} g_2$ is assumed to be intractable.

1. Alice randomly chooses $r \in \{1, ..., p\}$,
2. Alice calculates $C = \text{COM}(r, x) = g_1^r g_2^x \in G_p$,
3. Alice sends C to Bob.

To decommit, Alice sends r, x to Bob, who verifies that $C = g_1^r g_2^x$.

Theorem: The above $\text{COM}(\cdot, \cdot)$ satisfies the two requirements.

Proof: To show the first condition, i.e. that the scheme is information-theoretically secure, note that g_1^r, $r \in \{1, ..., p\}$, ranges uniformly over all elements of G_p. Consequently, every $C \in G_p$ is obtained as the value of $\text{COM}(x, r)$ with probability $\frac{1}{p}$.

To show the second condition, i.e. that the scheme is computationally binding, note that for Alice to reveal $x_1 \neq x$, she needs to compute r_1 and $x_1 \neq x$ such that $C = g_1^{r_1} g^{x_1}$, i.e. $g_1^r g_2^x = g_1^{r_1} g_2^{x_1}$, so $g_1^{r-r_1} = g_2^{x_1 - x}$. Note that $x_1 - x \neq 0$, so there is an inverse modulo p, i.e. $(x_1 - x)y = mp + 1$. Then, raising both sides of the equation to the power y gives $g_2 = g_1^{y(r-r_1)}$ by Fermat's little theorem. She has computed $\log_{g_2} g_1$, contradicting the assumed difficulty of the discrete logarithm problem.

Remark: It is possible to replace G_p with $(\mathbb{Z}_q)^*$, though choosing a generator g for $(\mathbb{Z}_q)^*$ is not always easy. The group of points over an elliptic curve can also be used.

Remark: Note that we have constructed a one-way function

$$f : \{0, 1\}^n \to \{0, 1\}^m,$$

where $m < n$, for which it is hard to find x_1 and x_2 for which $f(x_1) = f(x_2)$.

5.2 Rabin Cryptosystem

This is an encryption as safe as factorization. For the public key, Bob finds two large primes p, q (on the order of 2^{2048}) and computes $n = pq$. The private key is p (or q). To encrypt, Alice has message

$$x \in \mathbb{Z}_n^* = \{x : 1 \leq x < n, (x, n) = 1\}.$$

She encrypts it by computing $E_n(x) = x^2 \bmod n$.

Theorem/Algorithm: Using p, q, Bob can compute x up to four possible values.

Theorem: If Eve can decode messages $x^2 \bmod n = y$ in time T, then Eve can factor n in expected time $2T$.

Theorem: (Chinese remainder theorem) Let $a_1, a_2 \in \mathbb{N}$ such that $(a_1, a_2) = 1$. Let $a_1 > r_1$ and $a_2 > r_2$. There is a unique $m \leq a_1 a_2$ such that $m = r_1 \bmod a_1$ and $m = r_2 \bmod a_2$.

To decode, note that

$$\left(\frac{y}{p}\right) = 1; \qquad \left(\frac{y}{q}\right) = 1.$$

Then $\exists x_1^2 = y \bmod p$ and $x_2^2 = y \bmod q$. However, $(p - x_1)^2 = y \bmod p$ and $(p - x_2)^2 = y \bmod q$ as well. For each pair (x_1, x_2), $(p - x_1, x_2)$, $(x_1, p - x_2)$, and $(p - x_1, p - x_2)$, we can use the Chinese remainder theorem, we obtain four elements $z_i < n$.

Theorem: (Euler) We have

$$\left(\frac{y}{p}\right) = 1 \Leftrightarrow y^{\frac{p-1}{2}} \bmod p = 1.$$

Proof: Then there exists u such that $u^2 = y \bmod p$. Then

$$y^{\frac{p-1}{2}} = (u^2)^{\frac{p-1}{2}} = u^{p-1} \bmod p = 1 \bmod p.$$

Conversely, there are $\frac{p-1}{2}$ quadratic residues, and the polynomial equation

$$f(y) = y^{\frac{p-1}{2}} = 1$$

has at most $\frac{p-1}{2}$ solutions, so $f(y) = 1$ exactly at the quadratic residues.

Remark: In fact, $f(y) = -1$ if y is a nonzero quadratic nonresidue, i.e. we have

$$\left(\frac{y}{p}\right) = f(y) = y^{\frac{p-1}{2}}.$$

Let Bob choose $p = 4k_1 + 3$ and $q = 4k_2 + 3$. Let

$$\left(\frac{y}{p}\right) = 1 \Rightarrow y^{\frac{4k_1+3-1}{2}} = 1 \bmod p.$$

Then $x_1 = y^{k_1+1}$ so

$$x_1^2 = y^{2k_1+2} = y \cdot y^{2k_1+1} = y \bmod p,$$

and similarly $x_2 = y^{k_2+1} \bmod q$, so $x_2^2 = y \bmod q$.

Remark: It is possible to solve

$$x^n + a_1 x^{n-1} + ... + a_n = 0 \bmod \mathbb{Z}_p$$

efficiently using randomization.

6 Tuesday, September 20

6.1 The Rabin Cryptosystem

Algorithm: We have public key $n = pq$ and private key p. Now assume that $p = 4k_1 + 3$ and $q = 4k_2 + 3$. Alice has message $x \in \mathbb{Z}_n^*$. If $x \geq n$, break the message up into blocks and encrypt each block. Bob receives ciphertext $y = E_n(x)$, where $y = x^2 \mod n$. Using $n = pq$, Bob computes

$$x_1^2 = y \mod p; \qquad x_2^2 = y \mod p.$$

There are four possibilities $(\pm x_1, \pm x_2)$. By the Chinese remainder theorem, these give the four possible square roots of y in $\mathbb{Z}_n$. If $m < \log n$, and there is a fixed known prefix $x = wm$, then it is unlikely that two ciphertext have the given prefix. A better solution uses quadratic residuosity and the Jacobi symbol.

Theorem: If there exists an algorithm $\mathcal{A}$ such that given $y = x^2 \mod n$ computes in time T one of the $\sqrt{y} \mod n$, then using $\mathcal{A}$, n can be factored in expected time $2T$.

Proof: Consider $\mathbb{Z}_n^*$, where $|\mathbb{Z}_n^*| = (p-1)(q-1)$, so $4 \mid |\mathbb{Z}_n^*|$. Now

$$\mathbb{Z}_n^* = \{z_1^{(i)}, z_2^{(i)}, z_3^{(i)}, z_4^{(i)}\}, \qquad 1 \leq i \leq \frac{(p-1)(q-1)}{4},$$

where

$$(z_j^{(i)})^2 \mod p = y_i.$$

The squares modulo n in $\mathbb{Z}_n^*$ are

$$y_1, ..., y_{|\mathbb{Z}_n^*|/4},$$

since $x \mapsto x^2$ is a 4-to-1 mapping. Choose $x \in \mathbb{Z}_n^*$ uniformly randomly, and compute $y = x^2 \mod n$. Then y is uniform on $(\mathbb{Z}_n^*)^2$. Now compute $z = \mathcal{A}(y)$. If $x = z \mod p$ but $x \neq z \mod q$, then

$$x - \mathcal{A}(y) = 0 \mod p; \qquad x - \mathcal{A}(y) \neq 0 \mod q,$$

so $p \mid x - \mathcal{A}(y)$ and $p \nmid x - \mathcal{A}(y)$, so $\gcd(x - \mathcal{A}(y), n) = p$. Similarly, if $x \neq z \mod p$ but $x = z \mod q$, we can factor. Each of these happens with probability $\frac{1}{4}$, so the algorithm succeeds with probability $\frac{1}{2}$.

Theorem: The previous implies strong efficient authentication proocols, such as Fiat-Shamir.

6.2 RSA

Definition: The **Euler totient function** is

$$\phi(n) = |\mathbb{Z}_n^*|.$$

Example: Since $\mathbb{Z}_p$ is a field, we have $\phi(p) = p - 1$.

Proposition: $\phi(pq) = (p-1)(q-1)$.

Proof: This follows from the Chinese remainder theorem. Alternatively,

$$|\mathbb{Z}_n^*| = n - \frac{n}{p} - \frac{n}{q} + 1 = (p-1)(q-1).$$

Corollary: If $x \in \mathbb{Z}_n^*$, then $x^{\phi(n)} = 1 \bmod n$.

Algorithm: (RSA) To set up, Alice follows the following algorithm:

1. choose $n = pq$,
2. choose e uniformly randomly such that $(e, \phi(n)) = 1$,
3. compute d such that $ed = 1 \bmod \phi(n)$,
4. take public key to be (n, e) and secret key to be (n, d).

To encode $x \in \mathbb{Z}_n^*$, Bob calculates

$$c = \mathrm{Enc}_{n,e}(x) = x^e \bmod n.$$

To decode $c \in \mathbb{Z}_n^*$, Alice calculates

$$\mathrm{Dec}_{n,d}(c) = c^d = x^{ed} = x \bmod n,$$

since $ed = 1 \bmod \phi(n)$ and $x^{\phi(n)} = 1 \bmod n$.

Remark: If $p-1$ and $q-1$ are products of small primes, then RSA is breakable. Choosing p and q to be Sophie Germaine primes, then $p|q-1$ so the encryption is safe.

Remark: Pure RSA is malleable. If Alice knows $y = x^e \bmod n$, we can compute $(2x)^e = 2^e x^e \bmod n$.

6.3 Key Exchange

Algorithm: In general, a public key encryption (PKE) can be used to establish a common private key, say for AES. To do so, Bob looks up Alice's public key (n_A, e_A). Now Bob randomly chooses r, and sends Alice $c = \mathrm{Enc}_{n_A,e_A}(r)$. Alice decrypts $r = \mathrm{Dec}_{n_A,d_A}(r)$, so Alice and Bob now share the key r.

7 Thursday, September 23

7.1 Digital Signatures

Definition: Bob wants to use a **digital signature** to sign a document in a way that is unforgable. Let Bob have a secret signature key (ssk) d and a public signature verification key (psvk) e, and let $M \in \{0,1\}^*$

be a message. To sign M, Bob computes SIG(M). The pair $(M, \text{SIG(M)})$ is the signed document. Using e, the signature can be verified.

Remark: Digital signature requires a central key authority.

Definition: A hash function $H : \{0,1\}^* \to [0, n-1]$ is a function such that (i) $H(x)$ is easily computable, (ii) H is collision resistent, i.e. it is intractable to compute $M_1, M_2 \in \{0,1\}^*$ such that $H(M_1) = H(M_2)$.

Examples: SHA1, SHA2, MD-5. However, these have recently been shown not to satisfy property (ii). These can be computed efficiently (billion computations per second).

Algorithm: To encrypt a large message M, Bob can encrypt the Hash of M, i.e. use $(M, \text{SIG}(H(M)))$.

Algorithm: Now let $n = pq$. Bob uses ssk (p, q) and psvk n. Given $M \in [0, n-1]$ such that $(M, n) = 1$, Bob chooses x such that $x^2 = M \bmod n$. Then $\text{SIG}(M) = (M, x)$. Given n and (M, x), Alice can compute $x^2 \bmod M$ and check that this equals M. However, only $\frac{1}{4}$ of the elements of $\mathbb{Z}_n$ are squares. Pragmatically, we can assume $\log_2 M = \log_2 n - 10$, so we can assign a random prefix π such that πM has a square root, and give $(\pi \circ M, \sqrt{\pi \circ M})$ as the signed document.

Theorem: Computing SIG(M) for a given M is equivalent to knowing the factorization of $n = pq$.

Attack: To produce a signed document, Eve can select x and compute $y = x^2 \bmod M$. Then Eve can publish $\text{SIG}(y) = (y, x)$.

Definition: We say a digital signature is **existentially unforgeable** if Eve sees $\text{SIG}(M_1), ..., \text{SIG}(M_k)$. Then Eve cannot produce any Sig(M) where $M \neq M_i$ for $1 \leq i \leq k$.

Algorithm: Let $\log_2 m = \log_2 n - \delta$. Let π_B be a public prefix, and let $\log_2 \pi = \delta'$. Let $D = \pi \circ \pi_B \circ M$. Then $\text{SIG}(M) = (D, \sqrt{D} \bmod n)$.

7.2 RSA Digital Signatures

Algorithm: Bob has $n = pq$ and knows $\phi(n) = (p-1)(q-1)$. He produces (e, d) such that $ed = 1 \bmod \phi(n)$. Let the ssk be (n, d) and the psvk be (n, e). Let $M \in [0, n-1]$ be a message, and let $\text{SIG}(M) = (M, M^d \bmod n)$. To verify (M, x), check that $x^e = M \bmod n$.

Attack: As before, Eve can choose x and compute $(x^e \bmod n, x)$.

Algorithm: As before use a hash function H and a prefix makes RSA existentially unforgeable.

7.3 El Gamal Signatures

Proposition: Let $p = 2q + 1$, be a Sophie Germaine prime. The group $\mathbb{Z}_p^*$ is cyclic of order $p - 1$, and g is a generator if and only if $\left(\frac{g}{p}\right) = 1$.

Assumption: The discrete logarithm problem in $G = \mathbb{Z}_p^*$ is intractable.

Assumption: There exists a collision resistent hash function $H : \{0,1\}^* \to [[0, p-1]$.

Algorithm: Let $p = 2q + 1$ be a public Sophie Germaine prime. Also let H be a collision resistent hash function. Let the ssk be $x_A \in [0, p-1]$, chosen uniformly randomly, and let the psvk be (G, H, g, g^{x_A}). To sign $M \in \{0,1\}^*$, Bob follows the following algorithm:

1. choose $r \in [1, p-1]$ uniformly randomly and compute $h = g^r$,
2. compute $H(M \circ h)$
3. compute c such that
$$(g^{x_A})^{H(M \circ h)} g_c h = 1 \bmod p,$$
4. set $\mathrm{SIG}(M) = (M, h, c)$.

To verify the signature, Alice follows the following algorithm:

1. compute $H(M \circ h)$,
2. compute
$$(g^{x_A})^{H(M \circ h)} g^c h = 1 \bmod p$$
and checks that it equals 1.

Theorem: The algorithm is efficient.

Signing requires computing $h = g^r$, $H(M \circ h)$, c, and g^c. Verification requires computing $H(M \circ h)$, g^c, and $(g^{x_A})^{H(M \circ h)}$. To compute g^c, note that

$$1 = (g^{x_A})^{H(M \circ h)} g^c h = g^{x_A H(M \circ h) + c + r},$$

so we simply require that

$$x_A H(M \circ h) + c + r = 0 \bmod p - 1 \Rightarrow c = -(x_A H(M \circ h) + r) \bmod p - 1.$$

Theorem: Existential forgery of an El Gamal signature on a new M implies computing x_A from g^{x_A}.

8 Tuesday, September 27

8.1 El Gamal Signatures

Definition: A **random oracle function** is a "black-box" $f : D \to R$ such that given $x \in D$, $f(x)$ is chosen uniformly randomly from R. After $f(x)$ has been determined it is fixed.

Proposition: The probability of collision after n tries is

Algorithm: Let G be a group with $|G| = p$, let g generate G, and assume that the discrete logarithm problem is intractable in G. Let

$$H : \{0,1\}^* \to [0, p-1]$$

be a random oracle function. Bob chooses $\text{ssk}_B = x_B$ uniformly randomly from $[0, p-1]$, and publishes $\text{psvk}_B = g^{x_B}$. To sign $M \in \{0,1\}^*$, Bob follows the following algorithm:

1. choose r uniformly randomly in $[0, p-1]$,
2. compute $h = g^r$,
3. compute $H(M \circ h)$,
4. compute $(g^{x_B})^{H(M \circ h)}$,
5. compute $c \in [0, p-1]$ such that $(g^{x_B})^{H(M \circ h)} g^c h = 1$ by computing c such that

$$x_B H(M \circ h) + c + r = 0 \bmod p - 1.$$

Then

$$\text{SIG}_B(M) = (M, c, h).$$

To verify the signature, Alice follows the following algorithm:

1. compute $H(M \circ h)$,
2. compute $(g^{x_B})^{H(M \circ h)}$,
3. compute g^c,
4. verify

$$(g^{x_B})^{H(M \circ h)} g^c h = 1.$$

Remark: VeraSign uses an elliptic curve group EG of order p constructed by use of $\mathbb{F}_q$ where $q \tilde{2}^{200}$.

Lemma: If the discrete logarithm problem is intractable for G, then given $h_1, ..., h_k$, it is intractable to compute $m_1, ..., m_k \in \mathbb{Z}$ not all zero such that $h_1^{m_1} ... h_k^{m_k} = 1$.

Proposition: Signatures are not forgable.

Proof: Assume Eve has M. She chooses r and computes $h = g^r$ and then computes $H(M \circ h)$. She has to compute c such that

$$x_B H(M \circ h) + c + r = 0 \bmod p.$$

Then from c, r, and $H(M \circ h)$,

$$x_B = -\frac{c+r}{H(M \circ y)} \bmod p,$$

i.e. Eve has computed $x_B = \log_g g^{x_B}$. But the discrete logarithm problem is assumed to be intractable, a contradiction.

Theorem: Furthermore, the scheme is existentially unforgeable.

Remark: If signatures are constructed/verified as

$$(g^{x_B})^{H(M)} g^c h = 1,$$

then existential forgery is possible. In RSA with parameters n, d, and e_d, if $M_1^d \bmod n$ and $M_2^d \bmod n$, then $(M_1 M_2)^d \bmod n$ is computable.

8.2 Okamoto-Tanaka Key Agreement

Algorithm: Let $n = q_1 q_2$, where $q_i = 2p_i + 1$, and let

$$G = \{x \in \mathbb{Z}/n\mathbb{Z} : x = y^2\} = \mathrm{QR}_n.$$

Then G such that $|G| = p_1 p_2$. The key generation center (KGC) knows p_1, p_2, and $p_1 p_2$. The KGC chooses e and d such that $ed = 1 \bmod p_1 p_2$ and makes e public. Alice has $I_A \in \mathrm{QR}_n$ and Bob has $I_B \in \mathrm{QR}_n$. Alice securely communicates with the KGC to obtain $J_A = I_A^d \bmod n$, so $J_A \in \mathrm{QR}_n$. The public infrastructure consists of n, $g \in \mathrm{QR}_n$ a generator. For key agreement,

1. Alice chooses a uniformly randomly from $(\mathbb{Z}/n\mathbb{Z})^*$ and sends $J_A g^a$ to Bob,
2. Bob chooses b uniformly randomly from $(\mathbb{Z}/n\mathbb{Z})^*$ and sends $J_B g^b$ to Alice,
3. Alice computes

   $$(J_B g^b)^e = J_B^e g^{eb} = (I_B^d)^e g^{eb} = I_B g^{eb},$$

   and Bob similarly computes $(J_A g^a)^e = I_A g^{ea}$,
4. Alice divides by I_B to obtain g^{eb}, and Bob similarly computes g^{ea},
5. Alice computes $(g^{eb})^a = g^{eab}$, and Bob similarly computes g^{eab}.

Proposition: Assume that the Diffie-Hellman problem is hard for G. Then the above algorithm is resistent to man-in-the-middle attacks.

Proof: Note that Eve can sees $J_A g^a$, so she can compute g^{ea}, and similarly g^{eb}. She needs to compute Alice's key g^{eab}. From $(g^e)^a$ and $(g^e)^b$, to compute $(g^e)^{ab}$, which is the Diffie-Hellman problem.

9 Tuesday, October 4

9.1 Zero-Knowledge Proof

Definition: In an **interactive proof**, in round i,

1. the prover sends data d_i to the verifier,
2. the verifier sends challenge c_i to the prover,
3. the prover sends result r_i to the verifier.

Definition: An interactive proof is **complete** if a valid prover succeeds. It is **sound** if the probability that the verifier accepts when the prover does not know the secret is bounded by 2^{-k} after poly(k) iterations. It is zero-knowledge if there exists a simulator A that generates random exchanges $\{r_1, ..., r_k\}$ such that

$$\mathrm{P}(\{r_1, ..., r_k\}) = \mathrm{P}(A()).$$

Definition: In a **zero-knowledge proof**, the prover knows a secret, say the factorization of $n = pq$. He needs to prove to a verifier that he knows the secret without revealing any information. In other words, it is a complete, sound, and zer-knowledge interactive proof.

Algorithm: Assume n is public. Let $x \in [0, n-1]$ be a secret known by the prover, and let $y = x^2 \bmod n$. The prover proves that he knows $x = \sqrt{y}$ using the following algoritm:

1. the prover chooses $u \in \mathbb{Z}_n^*$ uniformly randomly,
2. the prover sends $w = u^2 \bmod n$ to the verifier,
3. the verifier chooses $c \in \{0, 1\}$ uniformly randomly,
4. if $c = 0$, the verifier asks for $\sqrt{w} \bmod n$, if $c = 1$, the verifier asks for $\sqrt{wy} \bmod n$,
5. if $c = 0$, then the prover sends u to the verifier, if $c = 1$, then the prover sends ux to the verifier,
6. if $c = 0$, then the verifier checks that $u^2 = w \bmod n$, if $c = 1$, then the verifier checks that $(ux)^2 = wy \bmod n$.

This process is repeated k times. If all responses are correct, then the verifier accepts the proof. Otherwise, the verifier rejects the proof.

Theorem: The algorithm is a zero-knowledge proof.

Proof: It is clear that the algorithm is complete. To show soundness, note that a prover succeeds with probability $\leq \frac{1}{2}$ unless he knows both $\sqrt{w}$ and $\sqrt{wy}$. Hence he knows $\frac{\sqrt{wy}}{\sqrt{w}} = \sqrt{y}$. After k tries, he succeeds with probability 2^{-k}. Finally, we claim zero-knowledge. Consider the following simulator:

1. choose $z \in \mathbb{Z}_n^*$,
2. choose $c \in \{0, 1\}$ uniformly randomly,
3. if $c = 0$, then he defines $w = z^2 \mod n$,
4. if $c = 1$, then he defines

$$w = \frac{z^2}{y} \mod n,$$

5. the simulator generates round $(d_i = w, c_i = c, r_i = z)$, which is correct.

It is that this exchange occurs with the same probability as an actual exchange.

10 Thursday, October 6

10.1 Simulations

Algorithm: The simulation follows the following algorithm to construct (d_i, c_i, r_i):

1. uniformly randomly choose $R \in \mathbb{Z}_n^*$ and $c \in \{0, 1\}$,
2. compute $w^{(0)} = R^2 \mod n$ and $w^{(1)} = R^2 y^{-1} \mod n$,
3. set $(d_i = w^{(c)}, c_i = c, r_i = R)$.

Conclusion: The simulator can construct a round r looking exactly like a round between the prover and the verifier.

Algorithm: Consider an exchange between a prover and a malicious verifier (w_i, c_i, r_i). He has a family of functions $C_i(r_1, ..., r_{i-1}, s)$, where $s \in \{0, 1\}^m$ is uniformly random. The malicious verifier does the following to simulate r_1:

1. uniformly randomly choose $R \in \mathbb{Z}_n^*$ and $c \in \{0, 1\}$,
2. compute $w^{(0)} = R^2 \mod n$ and $w^{(1)} = R^2 y^{-1} \mod n$,
3. compute $C_1 = C_1(w^{(c)}, s)$,
4. return $(d_1 = w^{(c)}, c_1 = c, r_1 = R)$ if $C_1 = c$, otherwise try again.

Note that the probability of success is $\frac{1}{2}$. To simulate r_i, after $r_1, ..., r_{i-1}$ were constructed, use the following algorithm:

1. uniformly randomly choose $R \in \mathbb{Z}_n^*$ and $c \in \{0, 1\}$,

2. compute $w^{(0)} = R^2 \bmod n$ and $w^{(1)} = R^2 y^{-1} \bmod n$,
3. compute $C_i = C_i(w^{(c)}, s)$,
4. return $(d_i = w^{(c)}, c_i = c, r_i = R)$ if $C_i = c$, otherwise try again.

Theorem: A malicious verifier cannot gain any information from an actual prover/verifier exchange that he cannot compute on his own.

Remark: The algorithm depends on the fact that the r_i are calculated successively. If the challenges are sent simultaneously, then the simulation succeeds with probability $\frac{1}{2^k}$, which is negligible. It is an open problem to prove that the algorithm is still zero knowledge.

10.2 Fiat-Shamir Authentication/Signatures

Assume the verifier wants to compute by means of $AL(\text{question}, \text{exchange}(P/V), s) = \text{answer}$, where $s \in \{0,1\}^m$ is uniformly random with probability of success ϵ. For example, question = factor n.

Algorithm: Assume user A has $y_A = x_A^2 \bmod n$. The user proves to the sattelite that he has knowledge of $\sqrt{y_A} \bmod n$. To do so, the user follows the following algorithm:

1. uniformly randomly choose $u_1, ..., u_k \in \mathbb{Z}_n^*$,
2. set $z = \text{username} \circ \text{time} \circ (w_1, ..., w_k)$, and send message $M = \circ H(z) \circ (R_1^{(c_1)} ... R_k^{c_k})$, where $H(z) = c_0 c_1 ... c_{2^m}$ is a hash and

$$R_i^{(\delta)} = \begin{cases} u_1 \bmod n \text{ if } \delta = 0 \\ u_1 x_A \bmod n \text{ if } \delta = 1 \end{cases} .$$

In practice, the first ten bits of the hash are random.

10.3 Authentication

Algorithm: Let n be public. Alice uniformly randomly chooses $x_1^{(A)}, ..., x_k^{(A)} \in \mathbb{Z}_n^*$, and computes $y_i^{(A)} = (x_i^{(A)})^2 \bmod n$ for $1 \leq i \leq k$, and her public information is $y_1^{(1)}, ..., y_k^{(A)}$. To authenticate, Alice follows the following algorithm:

1. uniformly randomly choose $u_1, ..., u_k \in \mathbb{Z}_n^*$,
2. compute $z_i = u_i^2 \bmod n$, for $1 \leq i \leq k$, and sends these to Bob,
3. Bob chooses a permutation $\sigma \in S_k$,
4. Alice computes $R_i = x_i u_{\sigma(i)} \bmod n$,
5. Bob verifies $R_i^2 = y_i z_{\sigma(i)} \bmod n$.

11 Tuesday, October 11

Review session on Wednesday during Varun's office hours.

11.1 Zero Knowledge Proofs for Membership in NP Languages

Definition: Let L be a language. Membership in L is NP if there is a polynomial time algorithm to verify that a given $w \in L$ is in fact in L.

Example: Membership in the set of 3-colorable graphs is NP.

Definition: Let L be a language. Membership in L is NP-complete if membership in any language can be reduced to membership in L.

Example: The langauge of all satisfiable propositional formulas, the language of all satisfiable propositions formulas in 3-conjunctive form, or the set of 3-colorable graphs.

Definition: A commitment function is a function $\text{COM} : \{0,1\}^m \to \{0,1\}^k$ such that

1. COM is computable in polynomial time,
2. (hiding) given $y = \text{COM}(x)$, computing x is intractable,
3. (binding) computing $x_1 \neq x_2$ such that $\text{COM}(x_1) = \text{COM}(x_2)$ is intractable.

We can also require that either of the last two properties be information-theoretic.

Example: (Pederson commitment) Let G be a group where $|G| = p$ is a large prime and where the discrete logarithm for G is intractable for every $e \neq g \in G$. Let $1 \neq g_1, g_2$ public. To commit to $x \in [0, p-1]$, randomly choose $r \in [0, p-1]$ and compute

$$\text{COM}(r, x) = g_1^r g_2^x.$$

This is information-theoretic hiding, since g_1^r is a uniformly random element of the group, so $g_1^r g_2^x$ is a uniformly random element of the group irrespective of x. If we can find $x_1 \neq x_2$ and $r_1, r_2 \in [0, p-1]$ such that $\text{COM}(r_1, x_1) = \text{COM}(r_2, x_2)$, then $\log_{g_1} g_2$ is computable.

Theorem: There is a zero-knowledge proof for G being 3-colorable.

Corollary: There is a zero-knowledge proof for membership in any NP language.

Proof: Let G be a graph, and let P know a 3-coloring for G. Let $G = (V, E) = (\{1, ..., n\}, \{e_1, ..., e_m\})$, where $e_k = (i, j)$, $i, j \in [1, m]$, $i \neq j$, is an edge, be a given graph. A coloring is a map $c : V \to \{1, 2, 3\}$, where $e = (i, j) \in E \Rightarrow c(i) \neq c(j)$. The prover and verifier follow the following procedure:

1. the prover randomly chooses $\sigma \in S_3$, and permutes the colors $\{1, 2, 3\} mapsto \{\sigma(1), \sigma(2), \sigma(3)\}$,
2. using the commitment function, the prover sends the verifier $(\text{COM}(\sigma(c(1))), ..., \text{COM}(\sigma(n)))$,

3. the verifier selects an edge $e \in E$ uniformly randomly,

4. the prover reveals $\sigma(c(i))$ and $\sigma(c(j))$,

5. the verifier verifies the commitments by computing $\text{COM}(\sigma(c(i)))$ and $\text{COM}(\sigma(c(j)))$ and checks that $\sigma(c(i)) \neq \sigma(c(j))$ and are in $\{1, 2, 3\}$.

If P does not know a 3-coloring, then

$$\text{P(not being caught)} \geq \frac{1}{|E|}.$$

11.2 Zero Knowledge Proofs for Auctions

Let bidders $B_1, ..., B_n$ give bids $x_1, ..., x_n$ to auctioneer. Assume bidder B_1 wins with bid x_1 and must pay x_2. The auctioneer must prove that he knows $x_1 > x_2 \geq x_3, ..., x_n (*)$. He computes and posts commitments $\text{COM}(r_1, x_1), ..., \text{COM}(r_n, x_n)$. The auctioneer wants to compute a circuit C with inputs $r_1, x_1, ..., r_n, x_n$ such that C verifies that $\text{COM}(r_i, x_i)$ is posted and that the inequalities $(*)$ hold. This becomes complicated, and a simpler method is required.

The algorithms will have input providers IP, evaluator prover EP, bidders B, and verifiers V. Let $\mathbb{F}_p$ be a field, where $p \sim 2^{128}$ is a moderate prime. The IP are $B_1, ..., B_n$, who provide inputs $x_1, ..., x_n \in \mathbb{F}_p$ to EP. Then EP performs a straight line computation (SLC) on the inputs. In other words, it is a sequence $x_1, ..., x_n, x_{n+1}, ..., x_L, x_{L+1}, ..., x_{L+k}$, where $x_1, ..., x_n$ are the inputs and $x_{L+1}, ..., x_{L+k}$ are the outputs. For every $m > n$, there exists $i, j < m$, $i, j \leq L$, such that

$$x_m = \begin{cases} x_i + x_j \bmod p \\ x_i x_j \bmod p \\ x_m = \text{TruthValue}(x_i \leq x_j) \text{ if } L < m \end{cases}.$$

The system uses a information theoretically hiding, computationally binding commitment function $\text{COM}(r, x)$. The IP submit $x_1, ..., x_n$ as $\text{COM}(r_1, x_1), ..., \text{COM}(r_n, x_n)$, and EP posts these on a secure bulletin board BB. Now EP performs the SLC and prepares a translation TR.

12 Thursday, October 13

12.1 Zero-Knowledge Proofs for Auctions

The IPs are the bidders $B_1, ..., B_n$ and the EP is the auctioneer. Consider a verifier V. Let $\mathbb{F}_p$ be a finite field, where $p \sim 2^{128}$. Now

1. the bidders $B_1, ..., B_n$ send bids $x_1, ..., x_n$ to the EP,

2. the EP performs a straight line computation (SLC), i.e.

$$x_1, ..., x_n, x_{n+1}, ..., x_L, x_{L+1}, ..., x_{L+k},$$

where

$$x_i = \begin{cases} \text{inputs, if } 1 \leq i \leq n \\ \text{intermediate steps, if } i < n \leq L \\ \text{outputs, if } L < i \leq L+k \end{cases},$$

and for all m, $n < m$, there exists $i, j < m$ $i, j \leq L$, such that

$$x_m = \begin{cases} x_i + x_j \\ x_i x_j \\ \text{TruthValue}(x_i \leq x_j) \text{ only if } m \geq L+1 \end{cases},$$

3. the EP announces the outputs,

4. upon demand, the EP provides a secrecy preserving proof of correctness, i.e. $x_1, ..., x_n, x_{n+1}, ..., x_L$ remain information theoretically secret.

Definition: Suppose there is $x \in \mathbb{F}_p$ and $X = (u, v)$. Define the **value** of X to be $\text{val}(X) = u + v$. If $\text{val}(X) = x$, then we say X **represents** x.

Algorithm: We construct a random representation $X = \text{RR}(x)$ by choosing $u \in \mathbb{F}_p$ uniformly randomly and setting $v = x - u$ and $X = (u, v)$.

Remark: Each coordinate of X is uniformly random.

Let $\text{COM}(r, x)$ be a commitment function that is information theoretically hiding and computationally binding.

Definition: Let $X = (u, v)$ A commitment $\text{COM}(X)$ is provided by choosing r, s uniformly randomly and computing

$$\text{COM}(X) = (\text{COM}(r, u), \text{COM}(s, v)).$$

To decommit X, reveal r, u, s, v. Verifying by receiver (say EP or the verifier).

Algorithm: Let bidder B_i have bid x_i.

1. B_i creates $X_i = \text{RR}(x_i)$,

2. B_i computes $\text{COM}(X_i) = (\text{COM}(r_i, u_i), \text{COM}(s_i, v_i))$ and sends this to the EP,

3. the EP posts these on a secure bulliten board,

4. at the end of the auction, B_i decommits by sending r_i, u_i, s_i, r_i to the EP,

5. the EP checks the commitment,

6. the EP now knows the input values x_i, and performs the known SLC and announces the output values $x_{L+1}, ..., x_{L+k}$,
7. the EP creates a translation TR of the SLC, i.e.

$$\text{COM}(X_1), ..., \text{COM}(X_n), \text{COM}(Y_1), ..., \text{COM}(Y_k),$$

8. the EP posts the TR on the bulliten board and posts claims such as $\text{val}(Y_1) = \text{val}(X_1) + \text{val}(X_2)$.

Algorithm: The following is an algorithm for the EP to give a zero knowledge proof of a claim. Let $\text{COM}(X_i) = (\text{COM}(r_i, u_i), \text{COM}(s_i, v_i)$ be posted, where $X_i = (u_i, v_i) = \text{RR}(x_i)$, for $1 \leq i \leq 3$. The prover claims $\text{val}(X_1) + \text{val}(X_2) = \text{val}(X_3)$, i.e. that $x_1 + x_2 = x_3$. This is true if and only if there exists $w \in \mathbb{F}_p$ such that

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} + \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_3 \\ v_3 \end{bmatrix} + \begin{bmatrix} w \\ -w \end{bmatrix}.$$

Now the EP does the following.

1. the prover reveals w,
2. the verifier picks $c \in \{0, 1\}$ uniformly randomly,
3. if $c = 0$, then the prover reveals u_1, u_2, u_3, and if $c = 1$, then the prover reveals v_1, v_2, v_3,
4. the verifier checks commitments and verifies that $u_1 + u_2 = u_3 + w$ if $c = 0$ and $v_1 + v_2 = v_3 - w$ if $c = 1$.

Theorem: The above algorithm is complete, sound, and zero-knowledge.

Proof: First we show completeness. If $x + y = z$, then the prover will succeed if either $c = 0$ or $c = 1$. Now we show soundness. If $x + y = z$ is false, and the prover claims it is true, then the prover will fail in at least one of the cases $c = 0$ or $c = 1$. Therefore the probability that the claim is accepted is $\leq \frac{1}{2}$. Finally, we show that the proof is zero-knowledge. The values revealed are completely independently random values, so this is clear.

13 Tuesday, October 18

Recall: We have an SLC, with input providers IP, evaluation prover EP, and a verifier V. We have $x, y, z \in \mathbb{F}_P$ with random representations $X, Y, Z \in \mathbb{F}_p^2$, respectively. We have commitments $\text{COM}(X)$, $\text{COM}(Y)$, and $\text{COM}(Z)$. Last time, we gave an information-theoretic zero-knowledge proof that $x + y = z$ given only X, Y, Z. Now we consider multiplication.

Algorithm: The EP claims that $\text{val}(X)\text{val}(Y) = \text{val}(Z)$, where $X = (u_1, v_1), Y = (u_2, v_2), Z = (u_3, v_3)$. Note that if the claim is true, then

$$u_3 + v_3 = u_1u_2 + v_1v_2 + u_1v_2 + u_2v_1.$$

The EP introduces $Z_1 = (u_1v_1, u_2v_2)$, $Z_2 = (u_1v_2+w_1, -w_1)$, and $Z_3 = (u_2v_1+w_2, -w_2)$ into the translation, where w_1 and $w_2 \in \mathbb{F}_p$ are uniformly random. Then

$$\text{val}(Z) = \text{val}(Z_1) + \text{val}(Z_2) + \text{val}(Z_3).$$

The translation now contains COM(X), COM(Y), COM(Z), COM(Z_1), COM(Z_2), and COM(Z_3). Now consider an interaction between the EP and the verifier. We have to break the interaction into various "aspects" that are independently verified.

In aspect 0, we verify that $\text{val}(Z) = \text{val}(Z_1) + \text{val}(Z_2) + \text{val}(Z_3)$ and $Z_1 = (u_1v_1, u_2v_2)$. The verification proceeds as follows:

1. the EP announces $w \in \mathbb{F}_p$ uniformly random such that $Z = Z_1 + Z_2 + Z_3 + \begin{bmatrix} w \\ -w \end{bmatrix}$,
2. the verifier randomly chooses $c \in \{1, 2\}$,
3. if $c = 0$, then the EP reveals u_1, u_2, and u_3, i.e. the first coordinates of X, Y, and Z, respectively, and the first coordinate of Z_1, Z_2, and Z_3, and if $c = 1$, the EP reveal the second coordinates.

In aspect 1, the EP reveals $\text{val}(Z_2) = u_1u_2$. In aspect 2: the EP reveals $\text{val}(Z_3) = u_2v_1$.

In global verification, the verifier chooses:

1. verify all additions and aspects 0, with probability $\frac{1}{2}$,
2. verify all aspects 1 with probability $\frac{1}{4}$,
3. verify all aspects 2 with probability $\frac{1}{4}$.

To verify aspect 1, the EP reveals u_1, v_2, and both coordinates of Z_2. Note that revealing both coordinates is okay because we are only revealing aspect 0. However, if the same x appears in two products, i.e. $x_1x_2 = y_1$ and $x_3x_1 = y_2$, this implies that both u_1 and u_2 are revealed. To solve this, for any X appearing both ways, the EP prepares X_R and X_L, proves in aspect 0 that $\text{val}(X_R) = \text{val}(X_L) = \text{val}(X)$, and uses X_L to prove one way and X_R to prove the other, i.e. X_L to prove that $x_1x_2 = y_1$ and X_R and to prove that $x_3x_1 = y_2$.

Theorem: Given one tranlsation, if even one claim is incorrect, then the probability that the verifier accepts is $\leq \frac{3}{4}$.

Proof: If the claim is false, then for some addition or aspect 0 of multiplication, the claim is false. The probability for the verifier not to accept is

$$\text{P[choosing aspect 0]P[choosing } c \in \{0, 1\} \text{ revealing falsehood]} \geq \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4}.$$

If the translation is false for aspect 1 (resp. aspect 2), if the coin toss says to choose aspect 1 (resp. aspect 2), then if the verifier will check aspect 1 (resp. aspect 2) and will not accept. Hence the probability that the verifier will accept is at most $\frac{1}{4}$ in either case.

Some outputs are of form $x_m = \text{TruthValue}(x_i \leq x_j)$. These will be only for $x_i, x_j < \frac{p}{32}$, where $p \sim 2^{128}$. To prove to the verifier that $x \leq y$, it suffices to show that $x, y, (y - x) < \frac{p}{2}$, where $y - x$ is taken modulo p.

Theorem: Let $b < \frac{p}{2}$ and $0 \leq x < b$. Then the EP can incorporate into the translation a proof that $0 \leq \text{val}(x) < 2b$ or that $p - b \leq \text{val}(x) < p$.

Corollary: $0 \leq \text{val}(Z^2) < 4b^2$.

Proof: We state the following two theorems without proof:

Theorem: (Lagrange) Every $x \in \mathbb{N}$ can be written as the sum of four squares.

Theorem: (Rabin) A representation can be computed in $O(\log x)^2$.

Let $X \in \mathbb{F}_p^2$ such that $\text{val}(X) = x$. We want to show that for $x < \frac{p}{32}$, then we can prove that $x < \frac{p}{2}$. Let $b^2 < \frac{p}{32}$. Then in the representation as a sum of four squares $x = z_1^2 + z_2^2 + z_3^2 + z_4^2$, we have $z_1, z_2, z_3, z_4 \leq b$, since $z_i^2 \leq x^2 < b^2$ for each i. The EP incorporates $\text{COM}(z_1)$, $\text{COM}(z_2)$, $\text{COM}(z_3)$, and $\text{COM}(z_4)$ into the translation. If the EP can give a zero-knowledge proof that $0 \leq \text{val}(z_i)^2 < 4b^2$ for each i, then he has proof that $\text{val}(X) \leq 4(4b^2) = 16b^2 < \frac{16p}{32} \leq \frac{p}{2}$.

14 Tuesday, October 25

Recall: We have $x \in \mathbb{F}_p$, IPs $P_1, ..., P_n$ with inputs $x_1, ..., x_n$, EP, and verifier V. The EP performs a straight line computation (SLC)

$$x_1, ..., x_n, ..., x_L, x_{L+1}, ..., x_{L+k},$$

translation

$$\text{COM}(x_1), ..., \text{COM}(x_n), \text{COM}(y_1), ..., \text{COM}(y_m).$$

Then EP can give an interactive proof with the verifier that the SLC is correct with probability at most $\frac{3}{4}$ of cheating.

Algorithm: Given $X = (u_1, v_1)$, $Y = (u_2, v_2)$ such that $\text{val}(X) = \text{val}(Y)$, with commitments $\text{COM}(X)$ and $\text{COM}(Y)$, the EP can prove $\text{val}(X) = \text{val}(Y)$. This is true if and only if there exists w such that $X = Y + (w, -w)$. To prove this claim, the EP does the following:

1. announce w,
2. the verifier chooses $c \in \{0, 1\}$ uniformly randomly,
3. if $c = 0$, then the EP reveals u_1 and u_2, and if $c = 1$ he reveals v_1 and v_2.

The probability that the verifier accepts on a false input is at most $\frac{1}{2}$.

Definition: Now suppose we have multiple commitments $\text{COM}(X_1), ..., \text{COM}(X_n)$ and $\text{COM}(Y_1), ..., \text{COM}(Y_n)$ such that $\text{val}(X_i) = \text{val}(Y_i)$. We say these two translations are **value consistent**.

Algorithm: To prove that two translations are value consistent, the EP repeats the above except announcing $w_1, ..., w_n$ simultaneously.

Algorithm: A k and L are chosen by the IPs, the EP, and the verifier V.

1. each IP submits $3k$ commitments $X_i^{(j)}$, where $1 \leq i \leq n$ and $1 \leq j \leq 3k$, of his input x_i, so we have $3k$ value consistent translations, these we denote by Y,
2. if one of the IPs cheat, then the EP will check and remove the IP from the auction,
3. now the Ep has all the commitments opened for him, so he knows $x_1, ..., x_n$ and proceeds to create $M = 11kL$ additional rows $\text{COM}(Y_i^{(j)})$, where $1 \leq i \leq n$ and $1 \leq j \leq m$, where we require that $\text{val}(Y_i^{(j)}) = x_i$, these we denote by Y
4. using X and $3kL$ rows of Y, the EP proves that
5. in X at least $2k$ rows are consistent (the values common to the above $2k$ rows are, by definition, the input values),
6. at most m/L (where $m = 11kL$) are **not** consistent with the $2k$ consistent rows of X from the previous step,
7. the EP has used all $3k$ rows of X and $5kL$ copies of Y, and uses the remaining $5kL$ rows of Y to give $5L$ interactive proofs.

14.1 El Gamal Digital Signatures

Recall: Consider a group $|G| = p$ where the discrete logarithm problem is intractable. Let $H : \{0,1\}^* \to [0, p-1]$ be a hash function, and let $g \in G$ be a generator. To set up, Alice does the following:

1. choose $\alpha \in [0, p-1]$ unfiromly ranndomly,
2. publish $g_1 = g^\alpha$ as public verification information, and keep α as the secret signature information.

To sign a document M, Alice does the following:

1. choose $h \in G$ uniformly randomly,
2. compute $s = H(M \circ h)$,
3. compute $g_1^{H(M \circ h)}$ and r such that

$$g_1^{H(M \circ h)} g_1^h g_1^r = 1,$$

   by solving

$$H(M \circ h) + h + r = 0 \bmod p - 1,$$

4. publish signature (M, h, r).

A verifier checks that $g^{H(M \circ r)} g^h g^r = 1$.

15 Thursday, October 27

15.1 El Gamal Signatures

We continue our discussion of El Gamal signatures. Let $g_1 = g^\alpha$, where α is the secret key. Recall that we are given signature (M_i, h_i, r_i) such that

$$h_i g_i^{H(M_i \circ h_i)} g^{r_i} = 1,$$

for $1 \leq i \leq k$.

Theorem: An adversary cannot create a signature (M, h, r) for any $(M, h) \neq (M_i, h_i)$.

In preparing the signature, Alice chooses $h_i \in G$ uniformly randomly. Note that

$$h_i g_1^{H(M_i \circ h_i)} g^{r_i} = 1 \Leftrightarrow h_i = g_1^{-H(M_i \circ h_i)} g^{-r_i} = g_1^{s_i} g^{t_i}.$$

Assume $H : \{0,1\}^* \to [0, p-1]$ is a random oracle hash function.

Definition: A **random oracle hash function** is a function

$$H(w) = \begin{cases} v \text{ if } H(w) \text{ already determined} \\ v \in [p-1] \text{ uniformly randomly otherwise} \end{cases}.$$

The adversary can produce (without α) sequences $\bar{h}_i = g_1^{\bar{s}_i} g^{\bar{t}_i}$, for $1 \leq i \leq k$, where $\bar{s}_i \in [0, p-1]$ is uniformly random and $\bar{t}_i \in [0, p-1]$. Then the adversary can compute $\bar{h}$ by the previous equation. This gives the same distribution as if the messages M_i were encoded using the encryption algorithm. Now the adversary has (M, h). Assume that he can produce a signature with nonnegligible probability, say at least $\frac{1}{2}$.

The adversary starts by randomly producing $\bar{h}_i$, for $1 \leq i \leq k$, as above. He tries to produce (M, h) different from all (M_i, h_i) that he is given. To do so, he asks for $H(M \circ h)$, and is able to find r such that $h g_1^{H(M \circ h)} g^r = 1$. Since H is a random oracle, he should be able to complete the signature for multiple values of $H(M \circ h)$, i.e. there exists $s' \neq s$ such that the adversary can find r' such that $h g_1^{s'} g^{r'} = 1$. Then $g_1^s g^r = g_1^{s'} g^{r'}$, which implies that the discrete logarithm can be computed.

15.2 Efficient Zero Knowledge Proofs

Recall that the IPs each gave the EP $3k$ commitments to his or her input value. Denote each of the $3k$ sets of commitments by R_i, for $1 \leq i \leq 3k$. The EP verifies for himself tha tthe $3k$ rows are pairwise value consistent. The EP prepares additional rows R'_j for $1 \leq j \leq M$, where $M = 11kL$, such that each R'_j is value consistent with every R_i, for $1 \leq i \leq 3k$. Now the EP wants to prove to a verifier V that all $3k + M$ rows are pairwise value consistent.

Theorem: The verifier can be assured that

1. at least $2k$ of the rows R_i are value consistent,

2. no more than $\frac{M}{L}$ rows of the rows R'_j are not value consistent with the $2k$ rows in the previous.

Proof: The verifier will use all $3k$ rows R_i. and $6kL$ rows out of R'_j.

1. The verifier randomly chooses $2L$ rows from the rows R'_j, say $R'_{j_1}, ..., R'_{j_{2L}}$, and challenges the EP to prove that R_1 is value consistent with each of these.

   Proof: If the EP claim is true, then all checks input value will turn out true. If more than $\frac{M}{L}$ of the rows R'_j are not value consistent with R_1, then

   $$\mathrm{P}(V \text{ accepts}) \leq \frac{1}{2} + \left(1 - \frac{1}{L}\right)^{2L} \approx \frac{1}{2} + \frac{1}{e^2} < \frac{3}{4},$$

   where the second possibility comes when all $R'_{j_1}, ..., R'_{j_{2L}}$ are value consistent with R_1, so

   $$\mathrm{P}(R_{i_j} \text{ value consistent with } R_1) \leq 1 - \frac{1}{L},$$

   and this happens $2L$ times, and the second possibility comes when not all of $R'_{j_1}, ..., R'_{j_{2L}}$ are value consistent with R_1, in which case the probability of V accepting the claim is $\frac{1}{2}$. Since this is repeated $3k$ times for $R_1, ..., R_{3k}$, if there are more than $\frac{M}{L}$ rows that are not value consistent, the probability of accepting is

   $$\mathrm{P}(V \text{ accepts}) \leq \left(\frac{3}{4}\right)^{3k}.$$

2. Assume that no more than $\frac{M}{L}$ rows R'_j are not value consistent with the majority. We want to show that unless at least $2k$ out of $3k$ rows R_i are value consistent with the given majority rows of R'_j in the previous,

   $$\mathrm{P}(V \text{ accepts}) \leq .$$

16 Tuesday, November 1

16.1 Zero-Knowledge Proofs

Recall: Let each IP send the EP $3k$ commitments, which the EP organizes into rows R_i, where $1 \leq i \leq 3k$. Then, with negligible probability of being cheated, the EP can prove to the verifier that out of $M = 11kL$, at most $\frac{M}{L}$ are not value consistent with the overwhelming majority.

Theorem: At most k rows of the rows R_i are not value consistent with the majority of the other $2k$ rows. The probability of being cheated is

$$\mathrm{P}(V \text{ cheated}) =$$

Proof: Assume that k rows R_i are not value consistent with the majority of the M rows (produced by the EP). Say R_1 is a bad row. In the interactive proof V chose $2L$ rows out of M rows and asked for verification of value consistency with R_1. Inconsistency will not be detected by V if

1. all $2L$ rows chosen from the M rows are consistent with R_1, which happens with probability

$$\mathrm{P}(\text{row } R'_j \text{ is value consistent with } R_1),$$

so

$$\mathrm{P}(2L \text{ choices give value consistent rows}) \leq \left(\frac{1}{L}\right)^{2L}.$$

(Note: the last item seems to be $\left(1-\frac{1}{L}\right)^{2L}$, but this is wrong.)

2. There exists a row in the choice that is not value consistent, but this is not detected, which happens with probability at most $\frac{1}{2}$.

Hence

$$\mathrm{P}(\text{happens } k \text{ times}) \leq \left(\frac{1}{2}+\left(\frac{1}{L}\right)^{2L}\right)^{k}.$$

16.2 Polynomial Interpolation

Let E be a field, and let $f(x) \in E[x]$ such that $\deg f = k-1$, and write

$$f(x) = a_{k-1}x^{k-1} + ... + a_0.$$

Let $\alpha_1, ..., \alpha_k \in E$, and let $f(\alpha_i) = \beta_i$ for $1 \leq i \leq k$.**Lemma:** A polynomial of degree $k-1$ over a field that is zero at k points is identically zero.

Theorem: If $g(x) \in E[x]$ such that $g(x) = k$ and $g(\alpha_i) = \beta_i$, for $1 \leq i \leq k$, then $g(x) = f(x)$.

Proof: Consider $h(x) = f(x) - g(x)$. Then $h(\alpha_i) = f(\alpha_i) - g(\alpha_i) = \beta_i - \beta_i = 0$ for $1 \leq i \leq k$. But $h(x)$ clearly has degree $k-1$, and it is 0 at k points. Hence $h(x) = 0$ so $f(x) = g(x)$.

Theorem: Given pairs (α_i, β_i) for $1 \leq i \leq k$, the interpolating polynomial $f(x)$ that can be efficiently computed.

Proof: Note that

$$\begin{bmatrix} 1 & \alpha_1 & ... & \alpha_1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_k & ... & \alpha_k^{k-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}.$$

Given that $\alpha_i \neq \alpha_j$ for $i \neq j$, this has a unique solution (by the Vandermonde determinant). Since inverse matrices are efficiently computable, the solution is efficiently computable.

16.3 Shamir Secret Sharing

Definition: In a **secret sharing scheme**, a dealer has secret $s \in \mathbb{F}_p$. He wants to share this amongst players $P_1, ..., P_r$ in a way that

1. any k players can can recover s,
2. any $k-1$ players have no information about s.

Algorithm: (Shamir secret sharing) Associate with players $P_1, ..., P_n$, where $n < p$, with values $\alpha_1, ..., \alpha_n \in \mathbb{F}_p$ such that $\alpha_i \neq 0$ and $\alpha_i \neq \alpha_j$ for all for all $1 \leq i < j \leq k$. Let $s \in \mathbb{F}_p$. To share s, the dealer does the following:

1. choose $a_1, ..., a_{k-1} \in \mathbb{F}_p$ uniformly randomly and set
$$f_s(x) = a_{k-1}x^{k-1} + ... + a_1x + s,$$
2. securely send share $s_i = f_s(\alpha_i$ to P_i, for $1 \leq i \leq k$.

Use interpolation to reconstruct s from k shares.

Theorem: The Shamir secret sharing scheme is correct and secure.

Proof: Let k players, say $P_1, ..., P_k$, attempt to reconstruct s. By polynomial interpolation, they can compute $f_s(x)$ and determine $f_s(0) = s$. This solution exists and is unique, so s is correctly reconstructed. To show that it is secure, consider any set of at most $k-1$ players. Without loss of generality, we have players $P_1, ..., P_{k-1}$. Let $s' \in \mathbb{F}_p$. By polynomial interpolation, there is a unique $f_{s'}(x)$ interpolating $f(\alpha_i) = s_i$ and $f(0) = s'$. Hence any value s' is consistent with $s_1, ..., s_{k-1}$, so no information is gained.

Remark: We have assumed that $P_1, ..., P_n$ are not malicious. Handling the case where a small number are malicious is much harder.

17 Thursday, November 3

17.1 Multi-party Computations

Definition: We have players $P_1, ..., P_n$ and dealer D with a secret s. An (n, k) **secret sharing scheme** shares s with the n players so that any k players can reconstruct s but no $k-1$ players can reconstruct s.

Definition: A **multi-party computation** allows players $P_1, ..., P_n$ with secrets $s_1, ..., s_n$, resp., to compute $s_1 + ... + s_n$ without revealing s_i for any i.

Algorithm: Assume dealers $D_1, ..., D_m$ have secrets $y_1, ..., y_m$ shared amongst $P_1, ..., P_n$ using
$$f_{y_i}(x) = a_{k-1}^{(i)}x^{k-1} + ...a_1^{(i)}x + y_i,$$
for $1 \leq i \leq m$. Then P_j has shares $s_j^{(i)} = f_{y_i}(\alpha_j)$, for $1 \leq i \leq m$ and $1 \leq j \leq n$. Then
$$\sum_{i=1}^{n} c_i s_j^{(i)},$$

where c_i is publicly known, is a secret sharing of

$$\sum_{i=1}^{n} c_i y_i.$$

To compute $y_1 y_2$, assume $n \geq 2k-1$. Now $P_1, ..., P_{2k-1}$ are commissioned to do the work. They hold $F(\alpha_j) = s_j^{(1)} s_j^{(2)} = f_{y_1}(\alpha_j) f_{y_2}(\alpha_j)$, for $1 \leq j \leq 2k-1$. Suppose that

$$F(x) = b_{2k-2} x^{2k-2} + ... + b_1 x + y_1 y_2.$$

Then

$$\begin{bmatrix} 1 & \alpha_1 & ... & \alpha_1^{2k-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{2k-1} & ... & \alpha_{2k-1}^{2k-2} \end{bmatrix} \begin{bmatrix} y_1 y_2 \\ \vdots \\ b_{2k-2} \end{bmatrix} = \begin{bmatrix} F(\alpha_1) \\ \vdots \\ F(\alpha_{2k-1}) \end{bmatrix}.$$

Denote the first matrix by B. Each player can compute $B^{-1} = [\beta_{i,j}]_{i,j=1}^{2k-1}$. Then

$$y_1 y_2 = \beta_{1,1} F(\alpha_1) + ... + \beta_{1,2k-1} F(\alpha_{2k-1}).$$

Now each P_j, for $1 \leq j \leq 2k-1$, uses an (n,k) secret sharing to share $\beta_{1,j} F(\alpha_j)$ amongst the players. To do this, P_j chooses a degree $k-1$ polynomial

$$f_{\beta_{1,j} F(\alpha_j)}(x) = \gamma_{j,k-1} x^{k-1}$$

and distributes shares

$$f_{\beta_{1,j} F(\alpha_j)}(\alpha_i)$$

to P_i. Player P_i obtains shares for $1 \leq j \leq 2k-1$, so P_i computes

$$f_{y_1 y_2}(\alpha_i) =$$

18 Tuesday, November 8

Recall: Let $P_1, ..., P_n$ be players. We showed that you could do an (n,k) multi-party computation, where $k < n$, of $y_1 + y_2 \bmod p$ or $y_1 y_2 \bmod p$. This is very powerful. Suppose you want to compute $y \oplus z$. This can be done by computing $(y-z)^2$.

18.1 Verifiable Secret Sharing

Suppose we have secret $s \in \mathbb{F}_p$, and the dealer (n,k) secret shares s by sending $s_i = f_s(i)$ to P_i, where $P_1, ..., P_n$ are players with $n < p$. We want to give a zero-knowledge proof to show that the secret has been shared, i.e. (i, s_i) lie on the graph of a polynomial of degree $k-1$. We assume that the dealer, along with at most $k-1$ of the players, are malicious, and that $n = 3k-1$. We also assume that the dealer or any player can broadcast information.

Algorithm: Recall the secret sharing algorithm. Given $s \in \mathbb{F}_p$, the dealer does the following to (n,k) secret share s:

1. choose $f_s(x) = a_{k-1}x^{k-1} + ... + a_1x + s$,
2. send $s_i = f_s(i)$ to P_i.

Algorithm: Now we give the desired zero-knowledge proof. Assume we have a group G such that $|G| = p$ and the discrete logarithm problem is intractable in G, and let $1 \neq g \in G$. The dealer and players do the following:

1. the dealer broadcasts g^s, $\{g^{a_1}, ..., g^{a_{k-1}}\}$, $\{g^{s_1}, ..., g^{s_n}\}$,
2. player P_i checks that

$$g^{s_i} = g^{s_i}$$

   and that

$$g^{s_i} = g^s g^{a_1 i} ... g^{a_{k-1} i^{k-1}},$$

3. if either equality fails, P_i broadcasts s_i,
4. if there are at most $k-1$ rejections, then the players reject the secret sharing.

Proof: Assuming the dealer is not malicious, ten

$$g^s g^{a_1 i} ... g^{a_{k-1} i^{k-1}} = g^{f_s(i)} = g^{s_i},$$

so P_i should accept. Since there are at most $k-1$ malicious players, all but $k-1$ will accept, so the players accept. In this case, the honest players will be able to reconstruct s. If the dealer is malicious, and there are at most $k-1$ protests, then there remain $2k$ unprotested shares. This may include the shares of the $k-1$ malicious users. Each player can verify that the share of another player is valid by performing the computation above. Since there are still $k+1$ honest players with unprotested shares, any k of them can reconstruct.

18.2 Time Lapse Cryptography

Recall: Consider an auctioneer and bidders $B_1, ..., B_n$ with bids $x_1, ..., x_n$, respectively, submitted using an encryption function $E(k_1, x_1), ..., E(k_n, x_n)$ before time T. At time T, each bidder B_i sends key $k_i$to the auctioneer. However, a bidder may refuse to reveal his key k_i, thus sabotaging the auction.

Definition: In **time lapse cryptography**, there are a number of players $P_1, ..., P_n$ and a sequence of pairs (e, T) of public keys e_i and times T_i. Under reasonable assumptions on the players, before time T_i, no player knows the decryption e_i, and after time T_i, the decryption key is unstoppably revealed.

Algorithm: Consider an (n, k) verfiable secret sharing. Every P_i has private signature key and public signature verification key. The players do the following:

1. P_i chooses secret $s_i \in \mathbb{F}_p$ and (n, k) shares it among the players,

2. for every secret s_i, perform the above verification,
3. every P_i digitally signs the secrets the player believes to be correct,
4. the secrets s_i having at least $2k$ signatures are definitely correct, say $s_1, ..., s_r$,
5. set the public key to be $e = g^{s_1}...g^{s_r}$,
6. the corresponding decryption key $s_1 + ... + s_r \mod p$ is reconstructed at time T by the honest players.

19 Thursday, November 10

19.1 Byzantine Agreement

Definition: A **Byzantine agreement protocol** (Lamport, Pease, Shostak 1980) is a protocol where players $P_1, ..., P_n$, at most k of which are malicious, have initial messages M_i. They exchange and update their messages in a way that

1. the exchange of message halts at some time t,
2. after time t, all the non-malicious players have the same message M',
3. if initially all non-malicious players have the same message M, then $M' = M$.

Malicious players can send any message to any player. We assume that if player P_i knows who he receives each message from, i.e. malicious players cannot impersonate other players.

Example: Consider n generals, at most k of whom are malicious. They will succeed if and only if at least $n - m$ of the generals attack. They start with message "attack" or "do not attack", and want to exchange messages until all the non-malicious generals have the same message, hence guaranteeing that if they attack, they will all attack.

Example: Consider a distributed database with n nodes, at most k of which are faulty, attempting to update an entry. The database wants to update only if all the non-faulty nodes update.

Theorem: There exists a Byzantine agreement protocol if messages are exchanged synchronously and at most $k - 1$ players become malicious during the protocol, where $k < \frac{n}{3}$. Furthermore, there does not exist a deterministic Byzantine agreement protocol in the case $k > \frac{n}{3}$.

Theorem: There does not exist a deterministic Byzantine agreement protocol if messages are exchanged asynchronously.

We will prove the following result:

Theorem: There exists a randomized Byzantine agreement protocol in the asynchronous case in (expected) fixed (independent of n and k) number of rounds as long as $k < \frac{n}{3}$.

Remark: The non-randomized algorithms take exponentially many rounds.

Remark: In the randomized case, it is actually possible for players to become malicious during the protocol.

Proof: For convenience, we will assume $k < \frac{n}{8}$. We need the following lemma:

Lemma: There exists a common true coin $c \in \{0, 1\}$, i.e. a uniformly random coin toss that is shared among all of the players.

Proof: (sketch) First (n, k) share the value of the coin. When it is time for the coin value to be revealed, the players reveal their shares. Any k non-malicious players can reveal the coin.

Now let P_i have variables

1. $m(i, T)$, the value of the message in round T,
2. $t(i)$, the temporary value of the message,
3. $c(i)$, a counter,
4. $\text{coin}(i)$, the value of the coin toss,
5. $R(i)$, the current round.

We have the following algorithm:

Algorithm: Consider player P_i, for $1 \leq i \leq n$. First P_i initializes $R(i) = 0$, $m(i, 1) = M_i$, $t(i) = \emptyset$, $c(i) = 0$, and $\text{coin}(i) = \emptyset$. In round T, for $T > 0$, P_i does the following:

1. set $R(i) = T$,
2. send $m(i, T - 1)$ to each other player,
3. wait until he receives $n - (k - 1)$ messages sent on round T,
4. set $t(i)$ to be the message occuring most frequently out of the received messages,
5. set $c(i)$ be the number of times $t(i)$ occured,
6. update $\text{coin}(i)$ for round T,
7. now set $m(i, T)$ as follows:
   (a) if $c(i) > \frac{n}{2}$ and $\text{coin}(i) = 0$, set $m(i, T) = t(i)$,
   (b) if $n - 2k \leq c(i)$ and $\text{coin}(i) = 1$, set $m(i, T) = t(i)$,
   (c) otherwise, set $m(i) = \emptyset$, i.e. the system is faulty,
8. end the turn.

20 Tuesday, November 15

20.1 Byzantine Agreement Protocol (BAP)

We have $p_1 \dots p_n$ players and $m_1 \dots m_n$ messages

1. k out of n may become improper $n > 6k$. BAP means that all proper generals will end up with the same message M.
2. If all proper players started with $m_i = M$ then their common agreement is on M.

If players have common random coin then Byzantine agreement possible in asyncronous case.

20.1.1 Common Coin: Via honest dealer D (dealer can be avoided but it is a pain)

D creates a sequence $c(0), c(1), \dots c(n)$ with $c(i) \stackrel{R}{\leftarrow} \{0,1\}$. He deals this among n players using $(n, n-2k)$ secret sharing. He gives each p_i a share $s(i,T)$ of $c(T)$,$1 \leq i \leq n$. Within round T of BAP, $n-2k$ players will exchange shares $s(i,T)$ and compute $c(T)$.

Every p_i has variables $m(i)$, $temp(i)$, and $count(i)$. In round 0:

1. p_i sets $m(i) = M_i$ where M_i is the initial message
2. $_i$ sends $(M_i, 0)$ to all and waits until receives $n-k$ values m_j including his m_i.
3. Sets $temp(i)$ values of received m_j's that constitutes a plurality
4. Sets $count(i)$ to number of times that temp(i) value was received.

p_i now (after updating $temp(i)$) sends (Ready,i,0) to all. "Lottery": Revealing/computing $c(i)$ (common coin for round 0) when/after p_i receives $n-k$ (Ready, i, round 0) messages. p_i sends $s(i,0)$ to all. Waits until he receives $n-k$ shares. Out of those $n-k$ shares at least $n-2k$ are from honest p_j's. How does p_i distinguish between honest/correct sgare and bogus ones sent by improper players?

Solution 1: D digitally signs shares.

Solution 2: Check vectors.

Now p_i computes $c(0)$ using $n-2k$ verified/good shares $s(j,0)$. Decision: If $count(i) > n/2$ and $c(0) = 0$ then update $m(i)$ to $temp(i)$. If $count(i) \geq n-2k$ and $c(0) = 1$ then $m(i) = temp(i)$. Else, if neither holds then make $m(i) =$ "$systemfault$". Note: The error message must be distinct from any possible starting message.

Theorem 1: If initially all proper p_i's had $m_i = M$ then in Round 0 and every subsequent round the end value $m(i)$ (at round T) $= M$

Assume that the proper p_i's do not have the same initial value.

Round T for p_i:
Every proper p_j enters round T with $m(j)$ as updated in Round $T-j$. Same steps as in Round 0. In decision for Round T if p_i (proper) has received $k+1$ messages $m(j) =$ "$systemfault$" then p_i sets $m(i) =$ "$systemfault$".

Theorem 2: For $t \geq 1$ the probablity that all proper players will emerge from Round T in agreement is $\geq 1/2$ (Pr(not in agreement) $\leq 1/2$)

Remark: Every proper player will reach and finish round T.

Proof: In round T $n-3k$ proper players need to get to Ready state and releasetheir shares for the coin $c(T)$ to be computable by the k improper players. The improper players cannot subvert the values of those $n-3k$ players.

Let these players be $p_{j_1} \ldots p_{j_{n-3k}} = S$. For every $p_j \in S$ $count(j) < n-2k$. If in this case $c(T) = 1$ then p_j will set $m(j) =$ "$systemfault$". In other words, in this case $p_j \in S$ will emerge with $m(j) =$ "$systemfault$"

21 Thursday, November 17

21.1 Wrapping up BAP

End game for BAP: p receives $n-k$ messages in round T then p_i releases his share and sends to all, $c(i,T)$ $n-2k$ shares received. Compute $c(T)$. It is absolutely enough to assume $k+1$ shares, $(n, k+1)$ secret sharing. Only now can $c(T)$ be created.

1. **Case 1:** Assume that p_i is a really fast to release his $C(i,T)$. He has $n-k$ messages from players claiming to be Ready, $n-2k$ $Ready_{j,T}$ messages from proper players. **None** of the p_j proper players had $count(i) > n-2k$. If $C(T) = 1$ according to decision, then every such player will set his $m(j)$ to "system fault". In next round, $T+1$, every proper player will recieve at least $n-2k$ messages $m(j)$ with "system fault". This gives $n-3k > k+1$ (Assuming $n \geq 6k+1$). p_i recieves "system fault" from **at least** the proper p_j's. Then, p_i sets his $m(i) =$ "$systemfault$".
2. **Case 2:** But if $n-2k$ there is (at least) one p_j who had $count(j) \geq n-2k$. Assume $C(T) = 0$. Consider some other **: proper** player p_l. Out of the $n-2k$ messages that p_j received with $temp(j) = \overline{M}$ p_l will recieve at least $n-3k$ wanted for $count(t) > n/2$.

Decision with $c(i)$ is created to set $m(l) = M$ (i.e. agreement at the end of round T for all proper players).

To remove dealer: $p_1 \ldots p_n$ can extend m, eat up values and do crazy things, but this is very compilcated and we won't go into it.

21.2 Homomorphic Encryption

$$c_1 \cdot E(c) \cdot c_2 = E(Y)$$
$$E(x) \cdot E(Y) \bmod n^2 = E(x+y)$$

21.2.1 Pailler Encryption

$n = pq$, p, q "large" primes. $p \nmid q-1$, $q \nmid p-1$. $\phi(n) = (p-1)(q-1)$, $\phi(n^2) = (p^2-p)(q^2-p)$

Fact from Euler: If $a \in \mathbb{Z}_n^*$ and $|G| = n$, $g^r = 1_G$ then $a^{\phi(n^2)}$

Let $\stackrel{r}{\leftarrow} [0, n-1], r \in \mathbb{Z}_n^*$

Equation 1: $C = E_n(x, r) = ((1+xn) \cdot r^n) \bmod n^2$

Note: r is the help value we use to encrypt.

Intractability Assumptions: Given any $y \in \mathbb{Z}_n^*$, it is intractable to determine if $\exists r$ s.t. $r^n \bmod n^2 = y$

The public key is $n(= pq)$, the private key is $\phi(n) = (p-1)(q-1)$

Encryption Bob has $x \in \mathbb{Z}_n$. Choose $r \stackrel{n}{\leftarrow} \mathbb{Z}_n^*$. Computes $C(x, r)$ and sends to Alice.

$$C^{M(n)} \bmod n^2 = ((1+xn) \cdot r^{n\phi(n)}) \bmod n$$

with $(r, n^2) = 1$ and $r^{n\phi(n)}$

Equation 2:

$$c^{M(n)} \bmod n^2 = (1+xn)^{(p-1)(q-1)} = (1 + (p-1)(q-1)xn + n^2) \mod n^2$$

$\exists u$ s.t. $u(p-1)(q-1) = an+1$. Multiply Eq. 2 by u. You get $u + (an+1)xn \bmod n^2 = (u + axn) \bmod n^2$. From that x can be computed.

$$C_1 = ((1+xn)r^n) \bmod n^2$$
$$C_2 = ((1+yn)s^n) \bmod n^2$$
$$C_1 \cdot C_2 \bmod n^2 = [1 + ((x+y) \mod n)^n (rs)^n] = E_n((x+y) \bmod n, (rs) \bmod n)$$

22 Tuesday, November 22

22.1 Pailler Encryption

Let $n = pq$ be the public key, $p \nmid q-1$ and $q \nmid p-1$, and let $\phi(n) = (p-1)(q-1)$ be the private key. Let $x \in [0, n-1] = \mathbb{Z}_n$. Let $r \in \mathbb{Z}_n^*$. Then

$$C = \text{Enc}_n(x, r) = (1 + xn)r^n \bmod n^2.$$

From C, using $\phi(n)$, x can be computed, from which r can be computed. First calculate

$$(1 + xn)^{-1}C \bmod n^2 = r^n \bmod n^2.$$

Since $(n, \phi(n)) = 1$, compute y and z such that $yn = 1 + z\phi(n)$. From $r^n = a + bn \bmod n^2$, where $0 \leq a < n$ and $(r, n) = 1 \Rightarrow a \neq 0$, compute

$$(r^n)^y = r^{1+\phi(n)z} \bmod n = r.$$

Now consider the mapping

$$\mathbb{Z}_n \times \mathbb{Z}_n^* \ni (x, r) \mapsto \text{Enc}_n(x, r) = (1 + xn)r^n \bmod n^2 \in \mathbb{Z}_{n^2}^*,$$

which is invertible, so it is 1-to-1. The number of pairs (x, r) is $|\mathbb{Z}_n||\mathbb{Z}_n^*| = n\phi(n) = \phi(n^2) = |\mathbb{Z}_{n^2}^*|$. Consequently, the mapping is also onto.

Assumption: Power n residiosity in $\mathbb{Z}_{n^2}^*$ is intractable, i.e. given $u, v \in \mathbb{Z}_{n^2}^*$, where one is a $r^n \bmod n^2$ and the other is not, it is intractable to compute which is which.

Definition: An encryption $\text{Enc}(x, r)$ is semantically secure if given an x in the domain of Enc and an encryption value C in the range of Enc, it is intractable to determine if C is one of the encryptions of x.

For example, consider pure RSA, where $C = \text{Enc}_{n,e}(x) = x^e \bmod n$. If an adversary sees C, and asks if C is the encryption of x. This is easy to answer.

Given $C = \text{Enc}(y, r)$, the adversary wants to determine if $y = x$ for some given x. Note that $\text{Enc}(0, t) = t^n \bmod n^2$, i.e. an nth power modulo n^2. The adversary computes $C_1 = \text{Enc}_n(x, s)$, where $s \in \mathbb{Z}_n^*$ is uniformly random. The adversary computes $C_1^{-1}C \bmod n^2 = \text{Enc}_n(x - y, s^{-1}r) \bmod n$. This equals $(s^{-1}r)^n \bmod n$ if and only if $y = x$. If it can be efficiently determined whether $y = x$, then it can be efficiently determined whether a given value is an nth power.

It is possible (Parkes, Rabin, Shieber, Thorpe 2007), to prove the correctness of announced second price auction, while keeping the bids secret, using Pailler encryption. Let $P_1, ..., P_k$ bid $x_1, ..., x_k \in [0, n-1]$. They submit $C_i = \text{Enc}_n(x_i, r_i)$ to the auctioneer. At the close of the auction, P_i sends x_i, r_i to the auctioneer. The auctioneer does not have $\phi(n)$. The auctioneer checks that $\text{Enc}_n(x_i, r_i) = C_i$. Now the auctioneer finds that $x_1 \geq x_2 \geq ... \geq x_k$, and announces P_1 the winner and tells him to pay x_2.

We use test sequences $2^m < n - 1$ and all bids $x_i < 2^m$. Test sequence $ts = (u_0, ..., u_{2m-1})$ a permutation of $(0, ..., 0, 1, 2, 4, ..., 2^{m-1})$ (m zeros). Note that $x \in [0, n-1]$ is $< 2^m$ if and only if there exists $u_{i_0}, ..., u_{i_{m-1}}$ in ts such that $x = u_{i_1} + ... + u_{i_{m-1}}$. The auctioneer has $C = \text{Enc}_n(x, r)$, and wants to prove to the verifier that $x < 2^m$. To prove it, the auctioneer prepares two test sequences $ts_i = (u_0^{(i)}, ..., u_{2m-1}^{(i)})$, it to

create $TS_i = (\text{Enc}_n(u_0^{(i)}, s_0^{(i)}), ..., \text{Enc}(u_{2m-1}^{(i)}, s_{2m-1}^{(i)}))$, where $i = 1, 2$. The auctioneer presents to the verifier $\text{Enc}_n(x, r)$, TS_1, and TS_2. Now V randomly chooses $c \in \{1, 2\}$. If $c = 1$, the prover opens all encryptions in TS_1 an the verifier checks that TS_1 constains a test sequence.

23 Tuesday, November 29

23.1 Oblivious Transfer

Definition: In an **oblivious transfer**, a sender S with a secret D, and a receiver R. Then R will get secret D with probability $\frac{1}{2}$, and S will not know whether R has gotten D.

Algorithm: We assume that the receiver R is computationally bounded. Then the sender S and the receiver R do the following:

1. S creates $n = pq$ and chooses $e \in \mathbb{Z}_n^*$ with corresponding RSA encryption function
$$\text{Enc}_{n,e}(x) = x^e \bmod n,$$
2. S sends
$$(n, e\text{Enc}_{n,e}(D))$$
to R,
3. R chooses $x \in \mathbb{Z}_n^*$ uniformly randomly and computes
$$y = x^2 \bmod n,$$
4. S computes $\bar{x}$ such that
$$\bar{x}^2 \bmod n = y,$$
5. S sends $\bar{x}$ to R,
6. R computes
$$(\bar{x} - x, n) \begin{cases} p \text{ with probability } \frac{1}{4} \\ q \text{ with probability } \frac{1}{4} \\ 0 \text{ with probability } \frac{1}{2} \end{cases},$$
7. if R receives p or q, i.e. the factorization of n, then R can decode $\text{Enc}_{n,e}(D)$ and get D.

Remark: We have the following theorems regarding the security of RSA:

Theorem: The lowest bit of D is secure under the RSA security assumption. Furthermore, if $D \in \{0, 1\}^n$, then $\log n$ bits are secure.

Remark: A malicious sender can send values of n that are not the product of exactly two primes. This can be fixed by requiring that S give a zero-knowledge proof to R that $n = pq$. Instead, the above works if $n = p^k q^h$, i.e. n has exactly two prime factors, which is an easier zero-knowledge proof.

23.2 Oblivious Selection

Definition: In an **oblivious selection**, a sender S has secrets D_0 and D_1. The receiver R selects D_i, for $i = 1, 2$, where S does not know i.

Algorithm: Say S and R use n-bit AES encryption $\text{AES}(K, D)$. Without loss of generality, say R's objective is to get D_0. Then S and R do the following:

1. S chooses $K_0 = x_1...x_n$ and $K_1 = y_1...y_n$,
2. S sends $\text{AES}(K_0, D_0)$ and $\text{AES}(K_1, D_1)$ to R,
3. S chooses $d_1...d_{nk}$ uniformly randomly in $\{0,1\}^{nk}$,
4. S uses the oblivious transfer protocol to send each d_i, for $1 \leq i \leq nk$, to R, so
$$\text{E}(\#d_i \text{ received}) = \frac{nk}{2},$$
so assume that this is the number of d_i received by R,
5. R makes two lists
$$R_0 = \{i_1, ..., i_{\frac{nk}{2}}\},$$
where R has received d_i for $i \in R_0$, and
$$R_1 = \{j_1, ..., j_{\frac{nk}{2}}\},$$
where R has not received d_i for $i \in R_1$,
6. R sends R_0 and R_1 to S,
7. S randomly partitions
$$R_0 = \bigcup_{m=1}^{n} R_m^{(0)},$$
and
$$R_1 = \bigcup_{m=1}^{n} R_m^{(1)},$$
where $|R_m^{(r)}| = k$ for each $1 \leq m \leq n$ and $r = 1, 2$, and $R_{m_1}^{(r)} \cap R_{m_2}^{(r)} = \emptyset$ for $m_1 \neq m_2$ and $r = 1, 2$,
8. S sends $R_m^{(i)}$ to R, for $1 \leq m \leq n$ and $i = 1, 2$,
9. S computes
$$\bar{x}_m = x_m \oplus \bigoplus_{i \in R_m^{(0)}} d_i,$$
and
$$\bar{y}_m = y_m \oplus \bigoplus_{i \in R_m^{(1)}} d_i,$$

10. S sends $\bar{x}_m$ and $\bar{y}_m$, for $1 \leq m \leq n$, to R.

Lemma: The receiver R will know $K_0 = x_1...x_n$ and will therefore get D_0.

Proof: By R's construction of R_0consisting of the nk indexes i for which he knows d_i, he knows all of the bits in $R_m^{(0)}$ for each $1 \leq m \leq n$, so he can decode $\bar{x}_m$ and get K_0.

Lemma: Regardless of how R splits $\{d_1, ..., d_{nk}\}$ into sets R_0 and R_1, where $|R_0| = |R_1| = \frac{nk}{2}$, under the above protocol,

$$\mathrm{P}(R \text{ will know a bit } x_i \text{ and } y_i) \sim \frac{1}{2^k}.$$

Proof: R knows d_i for $\frac{nk}{2}$ indices $i \in \{1, ..., nk\}$. In any partition $\{1, ..., nk\} = R_0 \cup R_1$, where $|R_0| = |R_1| = \frac{nk}{2}$, one of these contains at most $\frac{nk}{4}$ indices i such that R knows d_i, assume without loss of generality that this is R_1. Note that S randomly choose k indicees $i \in R_1$ to make $R_m^{(1)}$, so (approximately)

$$\mathrm{P}(R \text{ knows all } d_i) \leq \frac{1}{2^k},$$

so

$$\mathrm{P}(R \text{ knows some } y_j) \leq \frac{n}{2^k}.$$

To fix the proof, we need the following theorem due to Chernoff:

Theorem: Consider $X_i \in \{0, 1\}$ uniformly random, for $1 \leq i \leq n$. Let

$$X = \sum_{i=1}^{n} X_i.$$

Then

$$\mathrm{P}\left((1+\epsilon)\frac{n}{2} \leq X\right) \leq e^{-\frac{\epsilon^2 n}{2}},$$

and

$$\mathrm{P}\left((1+\epsilon)\frac{n}{2} \leq X\right) \leq e^{-\frac{\epsilon^2 n}{2}}.$$

24 Thursday, December 1

24.1 Millionaires Problem

Definition: In the **millionaires problem, two players P_1 and P_2 with secrets x_1 and x_2 respectively, want to compute TruthValue$(x_1 < x_2)$ without revealing x_1 or x_2 to one another.**

Theorem: There exists an algorithm to solve the millionaires problem.

24.2 Hyperencryption

Present encryptions depend on unproven assumptions, e.g. intractability of factorization. There are three ways to obtain provably secure encryptions:

1. **quantum encryption, but this requires expensive equipment and is limited in distance,**
2. **within the limited storage model,**
3. **within the limited access model.**

The last to items are hyperencryption. We will discuss the second method.

In an encryptionn $E_k(M)$, we want

1. **secrecy,**
2. **sender authentication,**
3. **message authentication,**
4. **non-malleability.**

We can achieve cryptographic security by computational security encryption is easy, decryption without the key is computational intractable, but this is unproven. It can also be achieved through information-theoretic security, a notion due to Shannon, where the ciphertext is essentially random.

Example: If Alice and Bob share a one-time pad $X = x_1...x_n$, where $x_i \in \{0,1\}$ uniformly random, then if Alice wants to share a message $M = m_1...m_n$, then $C = E_x(M) = M \oplus X$, and $M = \mathbf{Dec}_x(C) = C \oplus X = M \oplus X \oplus X = M$.

Remark: Note that this encryption is malleable. For example, an attacker can compute $C' = C \oplus 1^n$, flipping all the bits of the encrypted message M.

Remark: If a one-time pad is used multiple times, then each of these messages can be broken. Distribution is a difficult issue, since users must have a secure way to exchange the one-time pad.

24.3 Virtual Satellite

Consider $N = 50,000$ voluntary page server nodes (PSNs). Each PSN releases a random page only twice. The sender and receiver share a small random key K. Now

1. **using K, the sender and receiver anonymously and asynchronously download the same random pages from the same PSNs,**

2. **they XOR pages in groups of $k = 30$ to make one-time pads.**

Under the assumption that the adversary cannot compromise and monitor more than $M = 10,000$ PSNs simultaneously, we have

$$\mathbf{P}(\text{adversary has the one-time pad}) < \left(\frac{M}{N}\right)^k = \frac{1}{5^{30}}.$$

This is because the XOR is completely random to the adversary.

Remark: Each PSN P_i creates and replenishes random pages $P_i^{(j)}$, where $1 \leq j \leq 1000$. This can be done by extracting bits from a physical random source. Each $P_i^{(r)}$ is served only twice, and then removed and replaced.

Remark: Alice and Bob time-asynchronously share a stream of bits w_{2m}, w_{2m+1} to collect the "same" pages $P_i^{(j)}$. This stream is sourced by K and a portion of the subsequent shared random bits generated by the protocol. This is done by

1. **computing some function $i = F(w_{2m})$ to determine the PSN P_i to be used,**
2. **sending w_{2m+1} to P_i, who does the following:**
   (a) **keep track of a sequence of requests $u_1, u_2, ...$ for $P_i^{(1)}, P_i^{(2)}, ...$,**
   (b) **if $w_{2m+1} = u_j$ has already appeared, then P_i sends Alice $P_i^{(j)}$ and removes $P_i^{(j)}$ from the server,**
   (c) **otherwise, let j be the next unrequested page, set $u_j = w_{2m+1}$, and send $P_i^{(j)}$ to Alice.**

Remark: After Alice has collected unused pages $p_m, p_{m+1}, ..., p_{m+k}$, and Bob has collected unused pages $p'_{m'}, p'_{m'+1}, ..., p'_{m'+k'}$. Note that it may be the case that $p_j \neq p'_j$, $p_j = \emptyset$, or $p'_j = \emptyset$. Alice and Bob share a hash function h, and they do the following:

1. **Alice sends Bob $h(p_m), ..., h(p_{m+k})$,**
2. **Bob returns all $j \in [m, m+k]$ such that $p'_j \neq \emptyset$ and $\exists j' \in [m', m'+k']$ such that $h(p_j) = h(p'_{j'})$.**

One-time pads are used to encrypt reconciliation exchanges. Some of the randomness can be used to create fingerprinting polynomials (used to do pattern matching). If the pages are different, then the fingerprints are going to be different. If there is an adversary between Alice and Bob, then they Alice and Bob will discover this.

Remark: Alice and Bob XOR teh common reconciled common pages in groups of $k = 30$ to get common one-time pads. Three arrays of random bytes are created:

1. **one-time pads for Alice to send messages to Bob,**
2. **one-time pads for Bob to send messages to Alice,**

3. **bits for use by the system.**

The first two arrays of random bytes must be separate so that Alice and Bob do not synchronously attempt to use the same random bits to send messsages. The system bytes are used to choose the same random pages and to generate fingerprinting polynomials.

Remark: More simply but less securely, we can use

1. **use w_m to choose between a large number of webpages (e.g. through Google),**
2. **use w_{m+1} to select a random page,**
3. **use w_{m+2} to select a random linnk and get a page,**
4. **repeat the previous four times and add the final page to the trove,**
5. **extract randomness from collected pages,**
6. **perform reconciliation and then mix collected pages using XOR to get a one-time pad X.**