CS 220r: Cryptography, Notes

Jonathan Wang

Fall, 2009

CS 220r was taught by Professor Michael O. Rabin in the fall of 2009 at Harvard University.

Contents

[0] Contributions from Adrian Sanborn

1 Elementary Number Theory

9/8 Let $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\}$. An element $p \in \mathbb{N}$, $p \neq 1$ is *prime* if $a \mid p$ implies $a = 1$ or $a = p$ (e.g., $2^{400} - 593$ is prime). Let $\pi(x)$ denote the number of primes less than x.

The following theorem of Chebyshev shows that $\pi(x)$ grows like $\frac{x}{\ln x}$.

Theorem 1.1 (Chebyshev). $0.9 \cdot \frac{x}{\ln x} < \pi(x) < 1.1 \cdot \frac{x}{\ln x}$.

Theorem 1.2. *The integers $\mathbb{Z}$ are a unique factorization domain (UFD). That is, for $a \in \mathbb{N}$, one can write $a = p_1^{k_1} p_2^{k_2} \cdots p_\ell^{k_\ell}$, where $p_i \neq p_j$ for $i \neq j$, p_i are prime, $k_i \geq 1$. Furthermore if $a = q_1^{m_1} q_2^{m_2} \cdots q_t^{m_t}$ is another factorization, then $\ell = t$ and $q_1, \dots, q_t$ are a permutation of $p_1, \dots, p_\ell$ and the corresponding exponents are equal.*

For $a, b \in \mathbb{N}$, the greatest common denominator $\gcd(a, b)$ is the d such that

- $d \mid a$ and $d \mid b$, and
- If $d_1 \mid a$ and $d_1 \mid b$, then $d_1 \mid d$.

For a number n, let the "length" $\ell(n) = \lceil \log_2 n \rceil$. For example if $n \sim 2^{1000}$, then $\ell(n) = 1000$.

An algorithm AL on integers $a_1, \dots, a_m$ is *efficient* if there exists k such that $\mathrm{AL}(a_1, \dots, a_m)$ requires $\max(\ell(a_i))^k$ "steps".

Theorem 1.3. *Given n, determining if n is prime can be done efficiently.*

Conjecture 1.4. *It is an unproven assumption that there is no efficient algorithm for factorization.*

Theorem 1.5. *The $\gcd(a, b)$ can be efficiently computed.*

Proof. It is a fact (shown on the homework) that computing $a - b$ for $a > b$ is efficient. Write $a = 2^{k_1} a_1$, $b = 2^{k_2} b_1$ where $2 \nmid a_1, 2 \nmid b_1$. Then

$$\gcd(a, b) = 2^{\min(k_1, k_2)} \gcd(a_1, b_1).$$

Assume $a_1 > b_1$. Claim that $\gcd(a_1, b_1) = \gcd(a_1 - b_1, b_1)$.

($\Rightarrow$) Observe that $\gcd(a_1, b_1) = d \implies d \mid a_1$ and $d \mid b_1 \implies d \mid a_1 - b_1$ and $d \mid b_1$. I.e., d is a common divisor of $a_1 - b_1, b_1$.

($\Leftarrow$) If $d_1 \mid a_1 - b_1$ and $d_1 \mid b_1$, then $d_1 \mid a_1 - b_1 + b_1 = a_1$. Hence $d_1 \mid d$.

Observe that if $2 \nmid a_1$ or $2 \nmid b_1$, then $2 \nmid \gcd(a_1, b_1)$. Note that $a_1 - b_1$ is even. Consequently,

$$\gcd(a_1, b_1) = \gcd\left(\frac{a_1 - b_1}{2^m}, b_1\right) = \gcd(a_2, b_2).$$

(m is max power of 2) where $b_2 = b_1$ and $a_2 < \frac{a_1}{2}$. If $a_2 > b_2$ then above 2 steps reduced $\max(a_1, b_1)$ by $1/2$. Otherwise do again to get $(a_3, b_3) = (a_2, \frac{b_2 - a_2}{2^{m'}})$.

Replace (a_1, b_1) with (a_3, b_3), where $\gcd(a_3, b_3) = \gcd(a_1, b_1) = d$ and $\max(a_3, b_3) < \frac{1}{2}\max(a_1, b_1)$. Proceeding recursively we eventually get (a_ℓ, b_ℓ) where $a_\ell = b_\ell$ or $b_\ell = 0$. □

Division with remainder: Given a, n we can write $a = nq + r$ with $0 \leq r < n$. Let $a \bmod n := r$. We say that $a \equiv b \bmod n$ if $n \mid a - b$. We leave the following proposition as an easy exercise.

Proposition 1.6. *If $a_1 \equiv b_1 \bmod n$ and $a_2 \equiv b_2 \bmod n$, then $a_1 + a_2 \equiv b_1 + b_2 \bmod n$ and $a_1 a_2 \equiv b_1 b_2 \bmod n$.*

Multiplication, division, and taking remainder are all efficient algorithms. Therefore computing the product $a_1 \cdots a_4 \bmod n = (a_1 \bmod n) \cdots (a_4 \bmod n)$ can be done efficiently in length of n.

Proposition 1.7. *Let a, b, n each have at most k digits. Then we can compute $a^b \bmod n$ efficiently (in time polynomial in k).*

Proof. Algorithm: write $b = \epsilon_0 + \epsilon_1 \cdot 2 + \cdots + \epsilon_{k-1} \cdot 2^{k-1}$. Thus

$$a^b = a^{\epsilon_0}(a^{\epsilon_1 \cdot 2}) \cdots (a^{\epsilon_{k-1} \cdot 2^{k-1}})$$

where $\epsilon_i \in \{0, 1\}$. Now just calculate

$$x_1 = a,\ x_2 = a^2 \bmod n,\ x_3 = x_2^2 \bmod n,\ \ldots, x_{k-1}.$$ □

9/10 We proved the following proposition in the homework.

Proposition 1.8. *If $\gcd(a, b) = d$, then there exist $x, y \in \mathbb{Z}$ such that $ax + by = d$, $|x| < b$, and $|y| < a$.*

If $\gcd(a, b) = 1$, then we say that a and b are *relatively prime*. In this case there exist $x, y \in \mathbb{Z}$ such that $ax + by = 1$, which implies that $ax \equiv 1 \bmod n$. For example, $8 \cdot 2 \equiv 1 \bmod 15$.

For $1 < n \in \mathbb{N}$, define $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ and

$$\mathbb{Z}_n^* = \{a \mid 1 \leq a < n, \gcd(a, n) = 1\}.$$

For example, $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$.

The celebrated **Euler's totient function** is defined by $\varphi(n) := |\mathbb{Z}_n^*|$ (e.g., $\varphi(15) = 8$).

Proposition 1.9. *Let $n = pq$ for p, q prime. Then $\varphi(n) = (p-1)(q-1)$.*

Proof. Subtract numbers in $\mathbb{Z}_n$ that are divisible by p or q, then add back overlap: $pq - p - q + 1 = (p-1)(q-1)$. □

Proposition 1.10. *Consider $\mathbb{Z}_n^*$. Then $a, b \in \mathbb{Z}_n^*$ implies $(a \cdot b) \bmod n \in \mathbb{Z}_n^*$.*

Let $G = (G, \cdot)$ be a set with some binary operation. G is a group with respect to $\cdot$ if

1. $a, b \in G \implies a \cdot b \in G$. (Closure)
2. $a, b, c \in G \implies (a \cdot b) \cdot c = a \cdot (b \cdot c)$. (Associativity)
3. $\exists e \in G$ such that for $a \in G$, $a \cdot e = e \cdot a = a$. (Identity)
4. $\forall a \in G$, $\exists x \in G$ such that $a \cdot x = x \cdot a = e$. (Inverse)

If $|G| < \infty$, then G is finite. If $\forall a, b \in G$, $a \cdot b = b \cdot a$, then G is *commutative*.

Proposition 1.11. *$\mathbb{Z}_n^*$ is a commutative group with respect to multiplication mod n.*

Theorem 1.12 (Chinese Remainder Theorem)**.** *Let $\gcd(a_1, a_2) = 1$ and $r_1 \in \mathbb{Z}_{a_1}$ and $r_2 \in \mathbb{Z}_{a_2}$. Then there exists $x \in \mathbb{Z}_n$ ($n = a_1 a_2$) such that*

$$x \equiv r_1 \bmod a_1$$
$$x \equiv r_2 \bmod a_2$$

Furthermore x is unique modulo n.

Proof. There exist $u, v \in \mathbb{Z}$ such that $a_1 u + a_2 v = 1$. This implies that

$$a_1 u \equiv \begin{cases} 0 \bmod a_1 \\ 1 \bmod a_2 \end{cases} \qquad a_2 v \equiv \begin{cases} 1 \bmod a_1 \\ 0 \bmod a_2 \end{cases}.$$

Now let $x = r_1 a_2 v + r_2 a_1 u \bmod n$.

Uniqueness: suppose $\exists \bar{x} \in \mathbb{Z}_n$ with $\bar{x} \bmod a_1 = r_1$ and $\bar{x} \bmod a_2 = r_2$. Then $a_1 \mid \bar{x} - x$ and $a_2 \mid \bar{x} - x$, so since $\gcd(a_1, a_2) = 1$, this implies $n = a_1 a_2 \mid \bar{x} - x$. □

Let p be prime. Consider $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$. We say that $a \in \mathbb{Z}_p^*$ is a *quadratic residue (qr)* mod p if there exists $x \in \mathbb{Z}_p^*$ such that $x^2 \equiv a \bmod p$.

Lemma 1.13. *For $p > 0$ prime, the number of different qr* mod p *equals* $\frac{p-1}{2}$.

Proof. Consider $x^2 \bmod p$ for $1 \leq x \leq \frac{p-1}{2}$. We claim these are all distinct. Suppose $x^2 \equiv y^2 \bmod p$. Then $p \mid x^2 - y^2 = (x+y)(x-y)$. If both $x, y \leq \frac{p-1}{2}$, this is impossible. Thus there are at least $\frac{p-1}{2}$ quadratic residues. The rest of the squares are all of the form $(p-x)^2 \equiv x^2 \bmod p$ for $1 \leq x \leq \frac{p-1}{2}$, so we get nothing new. □

For $a \in \mathbb{Z}_p^*$, the **Legendere Symbol** is

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ q.r.} \\ -1 & \text{if } a \text{ quadratic non-residue} \end{cases}$$

Question 1.1. Can we compute $(\frac{a}{p})$?

Question 1.2. If $(\frac{a}{p}) = 1$, can we find an x such that $x^2 \equiv a \bmod p$?

2 $M^2 \bmod n$ encryption

Alice chooses two "large" primes $p, q \sim 2^{1000}$. Let $n_A = pq$ be the public key. The pair $\{p, q\}$ is the secret key.

Encryption: Bob has $M < n$ with $\gcd(M, n) = 1$. (If Bob found M not relatively prime to n, then he has factored n, and we assumed factorization is hard.)

$$\text{Bob} \xrightarrow{C = E_n(M) := M^2 \bmod n} \text{Alice}$$

C is the cipher text. Alice computes $C \bmod p = c_1$ and $C \bmod q = c_2$. Observe that $(\frac{c_1}{p}) = 1$ since $c_1 = M^2 \bmod p$. Similarly $(\frac{c_2}{q}) = 1$.

Alice can find[1] $x_1 \in \mathbb{Z}_p^*, x_2 \in \mathbb{Z}_q^*$ such that

$$x_1^2 \bmod p = c_1, \quad x_2^2 \bmod q = c_2.$$

She then finds an $M_1 \in \mathbb{Z}_n^*$ such that $M_1 \bmod p = x_1$ and $M_1 \bmod q = x_2$ by CRT. Now $M_1^2 \bmod p = c_1$ and $M_2^2 \bmod q = c_2$, so by uniqueness of CRT, $M_1^2 \bmod n = C$.

Note however that there are actually four distinct $M_1, M_2, \ldots, M_4$ modulo n such that $M_i^2 \bmod n = C$.

Proposition 2.1. *For $Y \in \mathbb{Z}_n^*$, the equation $Y \equiv X^2 \bmod n$ has exactly 4 solutions $X_1, X_2, X_3, X_4 \in \mathbb{Z}_n^*$, where $n = pq$ for primes p, q. Furthermore, we have* $\gcd(X_1 - X_2, n) = q, \gcd(X_1 - X_3, n) = p$.

Proof. When working modulo a prime, each quadratic residue has exactly two square roots. Hence, $X \bmod p$ is either equal to some $1 \leq x_p \leq \frac{p-1}{2}$ or $p - x_p$; similarly for q. Then each pair $(x_p, x_q), (p - x_p, x_q), (x_p, q - x_q), (p - x_p, q - x_q)$ yields a unique solution mod n by the CRT. Checking modulo p, q, CRT gives the last assertion. □

Fix for ambiguity: Bob adds a prefix $\bar{M} = \epsilon_1 \cdots \epsilon_{15} M$ where $\epsilon_1 \cdots \epsilon_{15}$ is specified in Alice's public key. Now cipher text is $C = \bar{M}^2 \bmod n$.

9/15 By the CRT, it is not hard to break the encryption given the factorization of n (just do the same thing Alice does).

[1] This is shown later for certain primes in Proposition 2.6.

Theorem 2.2. *Assume there is an algorithm AL where given an encryption $Y = X^2 \bmod n$ finds in time T a solution X_i such that $Y = X_i^2 \bmod n$. Then n can be factored in expected time $2T$.*

Proof. (Factoring Algorithm) Choose randomly $X \in [1, n-1]$. If $\gcd(X, n) \neq 1$ then you are done (the gcd must be p or q). Otherwise (in most cases) we have $X \in Z_n^*$. Compute $Y = X^2 \bmod n$. Compute $AL(Y) = X_1$ (assume it is the previously designated "first" solution WLOG). Then

$$(X - AL(Y), n) = \begin{cases} q & \text{if } X = X_2 \\ p & \text{if } X = X_3 \end{cases}$$

This yields a factorization and does so with probability $1/2$. Hence, expected time of algorithm is $2T$. □

Now suppose that AL computes the square root in 1 case out of 1000. To factor n: randomly choose X and run AL on $Y = X^2 \bmod n$ for time T. If AL succeeds, proceed as in the proof of Theorem 2.2; otherwise choose a new X and repeat. Then we can factor n in expected time $2000T$ instead.

We can therefore conclude that *decoding $M^2 \bmod n$ implies factorization of n*.

We now address how to actually find solutions to $y = x^2 \bmod p$.

Theorem 2.3 (Fermat's Little Theorem)**.** *For p prime, we have $a^p \equiv a \bmod p$ for any a.*

Proof. There are many ways to prove this. One way is to use induction. The case $a = 0$ is trivial. Now by the binomial theorem, we have

$$(a+1)^p \equiv a^p + \binom{p}{1} a^{p-1} + \cdots + \binom{p}{p-1} a + 1 \equiv a + 1 \bmod p,$$

where it is an easy exercise to show $p \mid \binom{p}{i}$ for $0 < i < p$. □

For $a \in \mathbb{Z}_p^*$, since a is invertible, Fermat's Little Theorem implies $a^{p-1} \equiv 1 \bmod p$.

A group is *cyclic* if there exists $g \in G$ such that $\{1, g, g^2, \ldots, g^{|G|-1}\} = G$. The element g is called a generator of G.

The proof of the following fact is slightly involved and uses algebra, so we omit it.

Lemma 2.4. *The multiplicative group of any finite field is cyclic. In particular $\mathbb{Z}_p^*$ is cyclic.*

Theorem 2.5 (Euler's criterion)**.** *An element $a \in \mathbb{Z}_p^*$ is a quadratic residue mod p if and only if $a^{(p-1)/2} \equiv 1 \bmod p$ (if not a qr, then it is -1). In other words, the Legendre symbol $\left(\frac{a}{p}\right) = a^{(p-1)/2} \bmod p$.*

Proof. If $a = x^2 \bmod p$, then $a^{(p-1)/2} \equiv x^{p-1} \equiv 1 \bmod p$ by Fermat's Little Theorem. For the other direction, $\mathbb{Z}_p^*$ is a cyclic group by Lemma 2.4. Let g be a generator and $a \equiv g^i \bmod p$. Clearly if i is even, then a is a qr. Since the order of g is $p-1$, we must have $g^{(p-1)/2} = -1$. Thus $a^{(p-1)/2} \equiv (-1)^i \bmod p$. If a is not a qr, i must be odd, so $(-1)^i = -1$. □

Proposition 2.6 (Algorithm)**.** *Suppose we are given a prime of the form $p = 4k + 3$. In this case we solve $x^2 = a \bmod p$ for x in the following way: Set $x = a^{k+1} \bmod p$. Then*

$$x^2 = a^{2k+1} \cdot a = a^{(p-1)/2} \cdot a \equiv a \pmod p$$

One-Time Pad

Note that in the $M^2 \bmod n$ cryptosystem above, encryption is pretty fast but decryption is a bit more computationally intensive. Suppose Alice has a powerful computer and Bob has a cell-phone processor. Then Bob can encrypt on his phone, send to Alice, and she can decrypt, but they can't send encrypted messages from Alice to Bob. To get around this, Bob encrypts the key to a one-time pad and sends it to Alice, and she uses that pad to encrypt her message and send it back.

Let $\oplus$ denote XOR. Alice and Bob have one-time pad X in common. Alice has message M and generates cyphertext $C = M \oplus X$. To decrypt, Bob calculates $C \oplus X = M \oplus X \oplus X = M$.

3 RSA encryption

3.1 Euler's Theorem

9/17 Take the prime factorization of $n = p_1^{k_1} p_2^{k_2} \cdots p_\ell^{k_\ell}$. It can be shown that

$$\varphi(n) = (p_1^{k_1} - p_1^{k_1-1}) \cdots (p_\ell^{k_\ell} - p_\ell^{k_\ell-1})$$

(one way is to use a probabilistic argument). In particular for $n = pq$, as before $\varphi(n) = (p-1)(q-1)$.

Theorem 3.1 (Euler's Theorem)**.** *If* $\gcd(a, n) = 1$*, then* $a^{\varphi(n)} \equiv 1 \bmod n$*.*

Proof. We will only prove the case $a^{(p-1)(q-1)} \equiv 1 \bmod n$ for $n = pq$. We have $a^{(p-1)(q-1)} \equiv (a^{p-1})^{q-1} \equiv 1 \bmod p$ since $p \nmid a$. Similarly, $a^{(p-1)(q-1)} \equiv 1 \bmod q$. Thus $a^{\varphi(n)} - 1$ is a multiple of p and q, so it is a multiple of pq.

For the general theorem, we have that $\mathbb{Z}_n^*$ is a group of order $\varphi(n)$. By Lagrange's Theorem, for any $a \in \mathbb{Z}_n^*$, we have $a^{\varphi(n)} \equiv 1 \bmod n$. □

3.2 Encryption system

Alice creates two large primes p, q. $n = pq$. Choose e such that

$$\gcd(e, \varphi(n)) = 1.$$

Let the public key be the pair $\text{pk}_A = (n, e)$. Alice computes d such that $ed \equiv 1 \bmod \varphi(n)$ by extended gcd algorithm. For this knowledge of $\varphi(n) = (p-1)(q-1)$ and in particular of the factorization p, q is needed. The secret key is $\text{sk}_A = (n, d)$.

Encryption (by Bob): for $1 \leq M < n$ with $\gcd(M, n) = 1$, the ciphertext is

$$C = E_{(n,e)}(M) = M^e \bmod n.$$

Lemma 3.2. *Let* $x \in \mathbb{Z}_n^*$*. We claim that* $(x^e)^d \equiv x \bmod n$*.*

Proof.

$$(x^e)^d \equiv x^{ed} \equiv x^{k\varphi(n)+1} \equiv x \bmod n.$$ □

Decryption: Alice has (n, d). By the lemma, $C^d \bmod n \equiv (M^e)^d \equiv M$.

RSA assumption: Decoding RSA without secret key is intractable. It is unknown whether breaking RSA implies factoring n. However, we have the following theorem.

Theorem 3.3. *Finding* d *such that* $(x^{ed-1}) \equiv 1 \bmod n$ *for all* $x \in \mathbb{Z}_n^*$ *implies factoring* n*.*

3.3 Semantic security [Goldwasser-Micali (1982)]

A class of encryption algorithms $\mathcal{E}$ is semantically secure if for any two messages $M_1 \neq M_2$, if we randomly choose $E \in \mathcal{E}$ and $\delta \in \{1, 2\}$, it is intractible, given $E(M_\delta)$, to compute if $\delta = 1$ or $\delta = 2$.

Let Alice be Broker and Bob be Client. Bob has M_1 = buy Google stock, M_2 = sell Google stock. Suppose we intercept

$$\text{Bob} \xrightarrow{E_{(n,e)}(M_\delta)} \text{Alice},$$

then if encryption isn't semantically secure, we can find out if B is buying or selling.

3.3.1 Pure RSA is NOT semantically secure

The interceptor can just compute $E_{(n,e)}(M_1)$ and $E_{(n,e)}(M_2)$, and compare them to C to find the answer.

To make RSA semantically secure, we use a *probabilistic encryption*. Alice specifies [in her public key] the length $\ell(P_1 \wedge P_2)$ and a prefix P_1. Bob randomly chooses $P_2 \in \{0,1\}^{100}$ and concatenates prefixes to get $X = P_1 \wedge P_2 \wedge M$. Assuming $X \in \mathbb{Z}_n^*$, the ciphertext is $C = E_{(n,e)}(X)$. Alice can then decode to get X, and since she knows the length of $P_1 \wedge P_2$. The adversary can't find which M_1, M_2 Bob is sending because he doesn't know the prefixes and there are too many possibilities to try. [Note: this does not totally preclude the possibility of other ways to distinguish M_1, M_2.]

3.4 Malleability

An encryption algorithm E is malleable if there is a function $f(M)$ and an efficient algorithm AL such that $AL(E(M)) = E(f(M))$. To see why this is bad, suppose originally we have

$$\text{Bob} \overset{C=E(M)}{\longrightarrow} \text{Alice}$$

There can be a man in the middle attack

$$\text{Bob} \xrightarrow{C} \text{intercept} \overset{\text{AL}(C)}{\longrightarrow} \text{Alice} \overset{\text{Decode}}{\rightarrow} f(M)$$

Now Alice gets the wrong message "from" Bob.

3.4.1 Pure RSA is malleable

Let $f(M) = uM \bmod n$. Then

$$(u^e \bmod n) \cdot (M^e \bmod n) = (uM \bmod n)^e \bmod n$$

Thus the person in the middle can change M to $2M \bmod n$.

To fix this, we use the same probabilistic encryption method mentioned before. If we have (prefix 1, prefix 2, M), then multiplying by $u^e \bmod n$ will change the prefix, which messes up the message. [Note: again, this does not show that other possible functions f do not work.]

4 ElGamal Encryption

Definition 4.1 (Sophie Germain Primes)**.** A number q is a Sophie Germain (SG) prime if there exists a prime p such that $q = 2p + 1$.

For example, 7 and 23 are SG primes.

It is an open problem whether there are infinitely many SG primes, but we will assume that SG primes of the size we need can be generated ($\sim 2^{100}$).

Let $q = 2p + 1$ be a SG prime. Define $QR_q = \{x^2 \bmod q \mid x \in \mathbb{Z}_q^*\}$.

Lemma 4.1. *Consider $G = (QR_q, \cdot \bmod q)$.*

1. $|G| = \frac{q-1}{2} = p$
2. *G is a cyclic group.* □

Proof. Since $\mathbb{Z}_q^*$ is cyclic, for a generator g, we have $\mathbb{Z}_q^* = g^{\mathbb{Z}}$ (slight abuse of notation). Then it is not hard to see that $GR_q = g^{2\mathbb{Z}}$. Note that $|G| = \frac{q-1}{2} = p$ is prime, so G is also cyclic. □

9/22 For a cyclic group $\bar{G}$ and any generator g, the discrete log problem for $(\bar{G}, g)$ is intractable if for $g_1 \in \bar{G}$, computing $\log g_1 = a$ such that $g^a = g_1$ is an intractable problem.

4.1 Computation Diffie-Hellman

Problem: $\bar{G}, g$. Given $g_1 = g^a$ and $g_2 = g^b$, compute g^{ab}.

CDH intractability assumption: the above is intractable.

Application of CDH intractability: Key exchange. Fast secret key encryption (say AES). Alice and Bob want to establish a one-time common AES key K. Alice randomly selects $a \in \{1, \ldots, |G| - 1\}$ and sends g^a to Bob. Bob randomly selects $b \in \{1, \ldots, |G| - 1\}$ and send g^b to Alice. Alice computes $(g^b)^a = g^{ab} = K$ and Bob computes $(g^a)^b = K$. Now by CDH intractability, an adversary knowing g^a, g^b cannot compute $g^{ab} = K$.

If discrete log can be efficiently computed for $\mathbb{Z}_q^*$, then CDH is tractable.

4.2 Decision Diffie-Hellman

Given $g_1 = g^a, g_2 = g^b$, and g_3, decide if $g_3 = g^{ab}$.

DDH intractability assumption: this problem is intractable.

4.3 Encryption system

We now describe the actual encryption system. Let $q = 2p + 1$, $p \sim 2^{1000}$, $\mathbb{Z}_q^*$. Alice randomly picks $a \in \{0, 1, \ldots, q - 1\}$ and a generator g of $\mathbb{Z}_q^*$. The secret key $\text{sk} = (q, g, a)$. Alice computes $h = g^a$. The public key is $\text{pk} = (q, g, h)$.

Remark: $q, \mathbb{Z}_q^*, g$ may be universal for a certain community of users.

Encryption: Bob has $M \in \mathbb{Z}_q^*$. He chooses random $b \in \{1, \ldots, q - 1\}$ and computes $g_1 = g^b$ and $h^b \cdot M$. The cipher text is $C = (g_1, h^b \cdot M)$. Note that $h^b = (g^a)^b = g^{ab}$.

Decryption: Alice has a and computes $g_1^{-a} = g^{-ab}$, where $-a$ is with respect to the order of $\mathbb{Z}_q^*$. Then $g^{-ab} h^b M = M$.

If an adversary knows $g^a = h$ and captured $(g^b, g^{ab} M)$, breaking the encryption means finding M, which is equivalent to computing g^{ab} from g^a, g^b. By assuming CDH intractability of $\mathbb{Z}_q^*$, breaking the ElGamal encryption is intractable.

Everything we have done so far works for any prime q.

4.3.1 Using $\mathbb{Z}_q^*$ does not give a semantically secure encryption

From $g^a = h$ and $g^b = g_1$, one can compute $\left(\frac{g^{ab}}{q}\right)$ using Euler's Criterion (Theorem 2.5). Say $g^{ab} \in QR_q$. Then $g^{ab} \cdot M \in QR_q$ iff $M \in QR_q$. Similarly if $g^{ab} \notin QR_q$, then $g^{ab} \cdot M \in QR_q$ iff $M \notin QR_q$ (this was a homework problem; we can reduce it to a question of parities). Thus if two messages differ in being qr, an adversary can determine which message it is.

To fix this, we use QR_q (for $q = 2p + 1$ SG) and messages $M \in QR_q$. Then CDH intractability holds for $G = QR_q$, which is a cyclic group of order p, and the resulting encryption is semantically secure [proof not given].

Drawback to ElGamal encryption: $\ell(M) = \log_2 q \sim 1000$. Now the ciphertext $C = (g_1 = g^b, h^b \cdot M)$, and $\ell(g_1) = \log_2 q$, $\ell(h^b \cdot M) = \log_2 q$. Therefore the ciphertext has twice the length of the message. In RSA, the ciphertext was approximately the same length.

5 Digital Signatures

Alice wants to "digitally" sign some document M with $SIG_A(M)$ such that:

1. Given M only Alice can produce $SIG_A(M)$.

2. Given $SIG_A(M)$ everyone can verify the signature.

There is a secret signature producing key ssk_A and a public signature verification key psvk_A.

5.1 RSA (pure) signatures

Alice produces p, q and $n = pq$. She also produces e, d as in the RSA encryption. She signs document M with

$$SIG_A(M) = (M, M^d \bmod n).$$

Let $\text{ssk}_A = (n, d, [p, q])$ and $\text{psvk}_A = (n, e)$. Now to verify, just check that $M \equiv (M^d \bmod n)^e \bmod n$.

Attacks: For any $Y \in \mathbb{Z}_n^*$, an adversary can compute $\bar{M} = Y^e \bmod n$ and post $(\bar{M}, Y) = SIG_A(\bar{M})$. Verifying will give $\bar{M} = Y^e \bmod n$, so anyone can sign documents as if they were from Alice, although they may not have control over what the documents say.

Malleable: Adversary can replace $SIG_A(M) = (M, M^d \bmod n)$ with

$$SIG_A(2^e M \bmod n) = (2^e M \bmod n, 2(M^d \bmod n)).$$

5.2 ElGamal signatures

9/24 A hash function H is an assignment

$$H : \{0,1\}^* \to \{0, 1, \ldots, n-1\} = [0, n-1]$$

such that

1. H is efficiently computable.

2. H is collision resistant, i.e., it is intractable to compute $x_1, x_2 \in \{0,1\}^*$ satisfying $x_1 \neq x_2$ but $H(x_1) = H(x_2)$.

Examples of such hash functions: SHA-1, SHA-2, MD-5.

There are also hash functors $H = H(-, r)$ that take random r as a parameter.

Let G be a cyclic group with generator g, $|G| = n$, where discrete log for G is intractable and computation of powers of g is efficient. E.g., let $q = 2p + 1$ SG prime, $G = \mathbb{Z}_q^*$, $|G| = q - 1$. Fact: any $g \in \mathbb{Z}_q^*$ such that $\left(\frac{g}{q}\right) = -1$ is a generator of $\mathbb{Z}_q^*$. This is proved in the homework.

Alice is the signer. The secret signature key $\text{ssigk}_A = (G, g, x_A \xleftarrow{R} [0, n-1], H)$, and public signature verification key is $\text{psigverk}_A = (G, g, g^{x_A}, H)$. Alice has $M \in \{0,1\}^*$. She signs using the following steps.

1. Pick random $y \in [0, n-1]$

2. Computes $h = g^y$.

3. Computes $H(M||h)$ where $||$ stands for concatenation.

[1] Zero knowledge proof of Sudoku: Take 81 cards and number them $1-9$ nine times. Flip the cards around and number the cards $1-81$. Collect cards $1, 10, 19, \ldots$ into box 1, cards $2, 11, 20, \ldots$ into box 2, etc. Now take all the cards out of one box and show the faces, which shows that all of $1-9$ appear in column 1. Now do this for columns, rows, and 3×3 squares.

4. Computes a $c \in [0, n-1]$ such that

$$(g^{x_A})^{H(M||h)} \cdot g^c \cdot h = 1_G. \tag{5.1}$$

To do this, note that by taking $\log_g$ of (5.1), we have

$$(x_A \cdot H(M||h) + c + y) \bmod n = 0.$$

Now we can easily compute c.

Alice signs M with $SIG_A(M) = (M, c, h)$.
Signature verification: Using psigverk$_A$, the verifier

1. Computes $H(M||h)$, computes g^c, and computes $(g^{x_A})^{H(M||h)}$.
2. Verifies (5.1).

Now we want to show that only Alice can sign messages. This signature is in a very strong sense unforgeable. A signature scheme is *existentially unforgeable* if an adversary who does not know the secret signature key of A, and who sees $\text{SIG}_A(M_1), \text{SIG}_A(M_2), \ldots, \text{SIG}_A(M_k)$ cannot produce $\text{SIG}_A(M)$ different from any of the above.

Note that if H is not collision resistant, Alice can sign something and then later claim she signed something else.

We assume that the hash function $H(-, r)$ is Black Box computable. Assume that random parameter r is included in ssigk$_A$ and psigverk$_A$. We slightly modify the signature process described above by using a new concatenation scheme. Pad M at the end so that $\ell(M) \geq \ell(h)$. Then let $M||^*h = M\varepsilon_1 \cdots \varepsilon_\ell h$ such that $\ell(M) = \ell + \ell(h)$, for some filler ε_i. This enforces a unique reading of $M||^*h$ (M is the first half, $\varepsilon_1 \cdots \varepsilon_\ell h$ is the second half).

Proof that El Gamal is existentially unforgeable. Assume adversary who saw the list of signed document can efficiently produce a new signature $\text{SIG}_A(M) = (M, c, h)$. New means $(M, h) \neq (M_i, h_i)$ for all $1 \leq i \leq k$. Now this implies $H(M||^*h) \neq H(M_i||^*h_i)$ for all $1 \leq i \leq k$ by non-collision. Adversary wants to sign M using an h such that $M||^*h \neq M_i||^*h_i$. Because of the black box nature of $H(-, r)$, he must have M, h to compute $H(M||^*h, r)$. Thus he is also able to forge using $H(M||^*h, r')$ for $r' \neq r$. The adversary produces $((g^{x_A})^{H(M||^*h,r)}, c, h)$ and $((g^{x_A})^{H(M||^*h,r')}, c', h)$ such that (5.1) holds. Dividing the two relations we get (h is canceled out)

$$g^{x_A(H(M||^*h,r) - H(M||^*h,r'))} \cdot g^{c-c'} = 1_G.$$

Since c, c' are known to the adversary who produced the signatures, taking $\log_g$ of the previous equation, x_A can be computed. This contradicts the intractability of discrete log for G, since all other information was independent of x_A. □

6 Zero Knowledge Proofs

Scenario: Alice knows a secret S. For example, given $n = pq$, Alice knows $\{p, q\}$. She wants to prove it to a verifier.

[GMR] introduced interactive proofs and zero knowledge proofs.

Interactive proof setup: Alice (prover) sends w_1 to verifier. Then verifier sends some random $c_1 \in \{0,1\}^*$ back as a challenge. Then Alice sends r_1 and verifier runs $V(w_1, c_1, r_1)$ which returns true/false. This completes Round 1. Then they do this for k rounds. The verifier accepts statement as true if all $V(w_i, c_i, r_i)$ are true for $1 \leq i \leq k$. For the interesting cases, V is assumed to be an efficiently computable function. So we limit the verifier, but the prover is not yet limited.

We say this interactive proof method is *complete* if when the prover knows S, the proof will be accepted every time. The method is *sound* if when the prover does not know S, the probability of success of accepting is $\leq \frac{1}{2^k}$.

6.1 Interactive ZKP of knowledge of $\sqrt{y} \bmod n$ [Fiat-Shamir]

9/29 Let $n = pq$ be public, with p, q unknown. Alice chooses random $x_A \in \mathbb{Z}_n^*$ and computes $y_A = x_A^2 \bmod n$. She wants to prove to a verifier given y_A that she knows x_A.

Exchange 1. Prover chooses random $u_1 \in \mathbb{Z}_n^*$ and sets $w_1 = u_1^2 \bmod n$. Prover sends w_1 to verifier. Verifier chooses random $c_1 \in \{0,1\}$ and sends to P. If $c_1 = 0$: give me $\sqrt{w_1} \bmod n$. If $c_1 = 1$: give me $\sqrt{w_1 y_A} \bmod n$. Now Prover responds by: if $c_1 = 0$, sends $r_1 = u_1$; if $c_1 = 1$, send $r_1 = u_1 x_A \bmod n$. This completes round 1, and the Prover and Verifier may repeat for as many rounds as necessary.

Completeness. If Prover knows a $\sqrt{y_A} \bmod n$, then will always be accepted.

Soundness. If Prover cannot compute a $\sqrt{y_A}$, then Pr(acceptance)$\leq \frac{1}{2^k}$ if the conversation lasts k rounds. With probability $1/2$ ($c_1 = 0$), Prover has to compute r_1 such that $r_1^2 \bmod n = w_1$. With probability $1/2$ ($c_1 = 1$), Prover has to compute $\bar{r}_1$ such that $\bar{r}_1^2 \bmod n = (w_1 y_A) \bmod n$. If the prover can correctly respond to both $c_1 = 0$, $c_1 = 1$, it means that in time T (say a minute), he computed r_1 such that $w_1 \equiv r_1^2 \bmod n$ and $\bar{r}_1$ such that $w_1 y_A \equiv \bar{r}_1^2 \bmod n$. Now by the *extraction process*

$$\left(\frac{\bar{r}_1}{r_1}\right)^2 \equiv \frac{w_1 y_A}{w_1} \equiv y_A \bmod n,$$

since everything is assumed to be in $\mathbb{Z}_n^*$. Conversely, if Prover cannot rapidly compute/know $\sqrt{y_A} \bmod n$, then Pr(passing Round 1)$\leq 1/2$, since Prover cannot pass both inquiries. Hence we have proved that:

Theorem 6.1. *If Prover cannot compute $\sqrt{y_A} \bmod n$ in time T, then*

$$\textit{Pr(acceptance of k-round conversation)} \leq \frac{1}{2^k}.$$

Zero-knowledge aspect of interactive proof. Claim: The Verifier does not learn anything from the interactive proof that he could not learn (with the same probability) on his own. To show this, we demonstrate that the Verifier can produce on his own conversations $(R_1, \ldots, R_k)$ with the same probability distribution as the conversations of the interactive proofs with the Prover. We consider a general Verifier who is not necessarily honest. In this case, the Verifier will choose c as a function of what he has already seen: during round i, he chooses $c(y_A; r_1, \ldots, r_{i-1}; w_1, \ldots, w_i)$.

Lemma 6.2 (Simulation algorithm)**.** *Given $c \in \{0,1\}$, the Verifier can produce a random pair $(w, r) \in (Z_n^*)^2$ such that if $c = 0$, then $w \equiv r^2 \bmod n$, and if $c = 1$, then $(w y_A) \equiv r^2 \bmod n$. The probability distribution of w is the same as if chosen by the Prover, and independent of c.*

Proof. The Verifier first chooses a random $r \in \mathbb{Z}_n^*$. If $c = 0$, set $w = r^2 \bmod n$. If $c = 1$, set $w = \left(\frac{r^2}{y_A}\right) \bmod n$. In either case, verification of (w, c, r) succeeds[2]. Since dividing by y_A is a bijection $\mathbb{Z}_n^* \to \mathbb{Z}_n^*$, for a fixed c the probability of any w being chosen is the same as choosing a random $r^2 \bmod n$, which is the probability distribution of the Prover. □

10/1 We now produce a simulation by a general Verifier inductively on the rounds. Assume that valid rounds $1, \ldots, i$ have already been constructed.

Simulating round $i+1$. The Verifier chooses a random $c \in \{0,1\}$ and then produces (w_{i+1}, r_{i+1}) using Lemma 6.2. V then sets $c_{i+1} = c(y_A; r_1, \ldots, r_i; w_1, \ldots, w_{i+1})$. Note that this choice of c depends on the just picked w_{i+1}. If $c_{i+1} = c$, then the Verifier uses the triple $(w_{i+1}, c_{i+1}, r_{i+1})$, which is a valid simulated exchange between Prover and Verifier. If $c_{i+1} \neq c$, then "rollback", i.e., pick a new c and repeat the process. By Lemma 6.2, we know that w_{i+1} and c are independent variables. Therefore the probability that $c_{i+1} = c$ is $1/2$. Thus generating round $i+1$ takes expected time 2.

[2]This method can be thought of as shooting a target *and then* drawing a bullseye around your mark, so it appears that you hit the bullseye.

Inductively producing the rounds, it takes expected time $2k$ to simulate k rounds. By Lemma 6.2, the w_i have same probability distribution as if picked by the Prover. Therefore the entire simulation has the same probability distribution as a conversation between the Prover and the Verifier. This shows the zero knowledge aspect.

The ZKP of knowledge of $\sqrt{y_A} \bmod n$ runs in sequential time $O(k)$ for Pr(cheating)$\leq \frac{1}{2^k}$.

Aside: we can consider replacing sequential conversation with "parallel" conversation where $w_1, \ldots, w_k$ are all sent at once, then $c_1, \ldots, c_k$ sent back, followed by $r_1, \ldots, r_k$ sent as a reply.

6.2 Non-interactive ZKP of knowledge of $\sqrt{y} \bmod n$

Scenario: Suppose n is public and the factorization $n = pq$ is not known. Alice chooses random $x_A \in \mathbb{Z}_n^*$ and calculates $y_A = x_A^2 \bmod n$. Alice makes y_A public. A TV satellite has a directory with (Alice,y_A), (Bob,y_B), (Charlie,y_C), etc. Each TV viewer has a satellite box. If Alice wants to watch a movie and pay for it, the satellite beams scrambled movie to her, Alice authenticates herself by proving knowledge of x_A, and the satellite send the unscrambling key. This must be done quickly.

Authentication process: Alice chooses random $u_1, \ldots, u_{10} \in \mathbb{Z}_n^*$. Set $w_i = u_i^2 \bmod n$. Let $M =$ (time stamp, name of program, $w_1, w_2, \ldots, w_{10}$). A hash function $H(M) = \varepsilon_1\varepsilon_2 \cdots \varepsilon_{128} \in \{0,1\}^{128}$. Alice sends the message: $(M, r_1, \ldots, r_{10})$ such that if $\varepsilon_i = 0$, then $r_i = u_i$; if $\varepsilon_i = 1$, then $r_i = (u_i x_A) \bmod n$. Receiver (satellite) computes $H(M) = \varepsilon_1\varepsilon_2 \cdots \varepsilon_{10} \cdots$ and verifies correctness:

$$r_i^2 \bmod n \stackrel{?}{=} \begin{cases} w_i & \varepsilon_i = 0 \\ (w_i y_A) \bmod n & \varepsilon_i = 1 \end{cases} \quad \forall i \in [1,10]$$

Suppose an adversary (cheater) creates $u_1^j, \ldots, u_{10}^j$ a thousand times and sets $w_1^j, \ldots, w_{10}^j$ to be the squares. From the hash function he can get $H(M^j) = \varepsilon_1^j \cdots \varepsilon_{10}^j \cdots$. Since this is random, there is a $\frac{1}{2^{10}} \approx 1/1000$ chance that $\varepsilon_1^j = \cdots = \varepsilon_{10}^j = 0$. Since the adversary does this 1000 times, he can pick the instance when this does happen. In this case he can authenticate himself with $u_1^j, \ldots, u_{10}^j$.

6.3 Digital signature using Fiat-Shamir

We can use the method described in the previous section to digitally sign documents. Take $\bar{M} \in \{0,1\}^*$. Alice picks random $u_1, \ldots, u_{20} \in \mathbb{Z}_n^*$ and computes $w_i = u_i^2 \bmod n$. Let

$$Z = (\text{Alice}, \bar{M}, w_1, \ldots, w_{20})$$

and compute $H(Z) = \varepsilon_1 \cdots \varepsilon_{20} \cdots \varepsilon_{128}$. The signature is then $SIG_A(\bar{M}) = (Z, r_1, \ldots, r_{20})$.

6.4 ZKP of knowledge of $\sqrt{y_1}, \ldots, \sqrt{y_{10}} \bmod n$

As in the original Fiat-Shamir scheme, assume $n = pq$ where p, q are unknown. Alice chooses random $x_1, \ldots, x_{10} \in \mathbb{Z}_n^*$ and sets $y_i = x_i^2 \bmod n$ for $i \in [1,10]$. Alice then makes $y_1, \ldots, y_{10}$ public. Alice authenticates herself to Bob by proving knowledge of $\sqrt{y_i} \bmod n$. This is a stronger version of the Fiat-Shamir scheme.

Lemma 6.3. *Suppose given $y_i \in \mathbb{Z}_n^*$, $i \in [1,20]$ which are quadratic residues mod n, one can efficiently compute, using an algorithm AL, for some $i < j$, the value $z_{ij} = \sqrt{\frac{y_i}{y_j}} \bmod n$. Then AL efficiently leads to factorization of n.*

Proof. Randomly choose $u_1, \ldots, u_{20} \in \mathbb{Z}_n^*$. Compute $y_1 = u_1^2 \bmod n, \ldots, y_{20} = u_{20}^2 \bmod n$ in time T. Apply $AL(y_1, \ldots, y_{20}) = (i, j, z_{ij})$ so that $z_{ij}^2 \bmod n = (y_i/y_j) \bmod n$. We also know that $(u_i/u_j)^2 \bmod n =$

$y_i/y_j \bmod n$. Therefore since u_i/u_j is random,

$$\gcd\Big(\frac{u_i}{u_j} - z_{ij} \bmod n, n\Big) = p \text{ or } q$$

with probability $1/2$ by Proposition 2.1. □

10/6 Assume there exists some commitment $COM()$ function [see Section 11]. The symmetric group S_{10} consists of all permutations π of the set $\{1,\ldots,10\}$. Bob (verifier) chooses $\pi \in S_{10}$ randomly (do this by choosing randomly from $[1,10]$ without replacement). Bob sends $COM(\pi)$ to Alice (prover).

Alice randomly chooses $u_1,\ldots,u_{10} \in \mathbb{Z}_n^*$, sets $w_i = u_i^2 \bmod n$, and sends $w_1,\ldots,w_{10}$ to Bob. Bob sends π as a challenge for Alice to show $\sqrt{y_1 w_{\pi(1)}},\ldots,\sqrt{y_{10} w_{\pi(10)}} \bmod n$. Bob cannot change π based on the w_i he sees because he is already committed by $COM(\pi)$. Alice now sends back $r_i = x_i \cdot u_{\pi(i)} \bmod n$. Bob verifies $r_i^2 \equiv y_i w_{\pi(i)} \bmod n$. Commitment is needed for zero knowledge.

Completeness. If P knows $\sqrt{y_1},\ldots,\sqrt{y_{10}} \bmod n$, then P will always be accepted by V.

Soundness.

Theorem 6.4. *If an adversary AD can for random $y_1,\ldots,y_{10}$ squares mod n pass the test (be accepted) then AD can factor n.*

Proof. If AD cannot respond correctly (in time T) to any $\pi \in S_{10}$, he is dead. Assume there exists one $\bar{\pi} \in S_{10}$ for which he responds correctly in time T.

$$Pr(\pi \text{ chosen by V } = \bar{\pi}) = \frac{1}{10!} \approx \frac{1}{3,600,000}.$$

Assume that there is another $\pi \in S_{10}$ such that AD can respond correctly to both $\bar{\pi}$ and π.

Claim: In this case $\exists \ell, m,\ \ell \neq m$ so that AD can compute in time $2T$ a z such that $z^2 \equiv (y_\ell/y_m) \bmod n$. This would imply that AD can factor n by Lemma 6.3.

Proof of claim: AD can compute in time T the square roots

$$\sqrt{y_1 w_{\pi(1)}},\ldots,\sqrt{y_{10} w_{\pi(10)}} \bmod n$$

(the correct response to challenge π). Also AD can compute in time T the square roots

$$\sqrt{y_1 w_{\bar{\pi}(1)}},\ldots,\sqrt{y_{10} w_{\bar{\pi}(10)}} \bmod n$$

(the correct response to challenge $\bar{\pi}$). Since $\pi \neq \bar{\pi}$, we also have $\pi^{-1} \neq \bar{\pi}^{-1}$. Thus there exists $k \in [1,10]$ such that $\pi^{-1}(k) \neq \bar{\pi}^{-1}(k)$. Since AD knows $\sqrt{y_{\pi^{-1}(k)} w_k} \bmod n$ and $\sqrt{y_{\bar{\pi}^{-1}(k)} w_k} \bmod n$, AD also knows by division

$$\sqrt{\frac{y_{\pi^{-1}(k)}}{y_{\bar{\pi}^{-1}(k)}}} \bmod n.$$

This proves the claim. We conclude that for random $y_1,\ldots,y_{10}$, Pr(AD passes test)$\leq \frac{1}{10!} \approx \frac{1}{3.6\times 10^6}$. □

10/8 *Zero knowledge aspect.*

Theorem 6.5. *An adversary AD cannot learn to authenticate himself even when after observing any number of authentications by Alice.*

Proof. AD observes actual interactive authentications Alice $\rightleftarrows$ Server (Verifier). Now we simulate authentications with the same probability distribution.

1. AD knows Alice's public $y_1,\ldots,y_{10}$.

2. AD chooses a $\pi \in S_{10}$ (with same distribution as verifier).
3. AD computes $COM(\pi)$.
4. Randomly chooses $r_1, \ldots, r_{10} \in \mathbb{Z}_n^*$.
5. Set $w_{\pi(i)} = r_i^2 / y_i \bmod n$.
6. Now the exchange $(w_1, \ldots, w_{10}; \pi; r_1, \ldots, r_{10})$ is valid.

As in the original Fiat-Shamir ZKP, we note that the w_i have the same probability distribution as if chosen by the Prover. Therefore observing real authentications gives no information to the adversary. □

6.5 A computationally efficient bijection $S_k \cong [0, k! - 1]$

Let $0 \leq m \leq k! - 1$. Assume $(k-1)! \leq m$ and write as $m = r_1(k-1)! + m_1$ for $r_1 \leq k-1$ and $m_1 < (k-1)!$. Repeating, we have

$$m = r_1(k-1)! + r_2(k-2)! + \cdots + r_{k-1}$$

where $0 \leq r_{k-1} \leq 1, \ldots, 0 \leq r_1 \leq k-1$. We map $m \in [0, k! - 1] \mapsto (r_1, r_2, \ldots, r_{k-1})$. We construct a permutation in S_k from $(r_1, \ldots, r_{k-1})$. Start with a 1. We can put 2 on the left or right, which we decide based on $r_{k-1} = 0$ or 1. Now we have three places to put 3, which we decide based on what r_{k-2} equals. Continuing, we get some permutation of $1, \ldots, k$ at the end. It is easy to see that this gives a bijection $S_k \cong [0, \ldots, k! - 1]$.

6.6 Non-interactive ZKP of knowledge of $\sqrt{y_1}, \ldots \sqrt{y_{10}} \bmod n$

Alice picks random $u_1, \ldots, u_{10} \in \mathbb{Z}_n^*$. Computes $w_i = u_i^2 \bmod n$. Now the hash function

$$H(y_1 y_2 \cdots y_{10} w_1 \cdots w_{10}) = \varepsilon_1 \varepsilon_2 \cdots \varepsilon_{128} \mapsto \pi \in S_{10}$$

where we truncate/mod the hash value to get a number in the interval $[0, 10! - 1]$ and use the bijection $[0, 10! - 1] \cong S_{10}$ given in the last section. Now Alice posts a signature of $(y_1, \ldots, y_{10}; w_1, \ldots, w_{10}; r_1, \ldots, r_{10})$ as the correct response to π.

7 Overview of Financial Cryptography

7.1 Time Lapse Cryptography

10/20 Time lapse cryptography service (TLCS) publishes data of time T_1, encryption key e_1 (public encryption key), T_2, e_2, etc. At time T_1, the TLCS publishes d_1, the corresponding decryption key. Before time T_1, no one knows d_1. After that time, no one can stop the corresponding item from being decrypted.

7.2 Vickery auctions

At a sealed bid auction, there are bidders $B_1, \ldots, B_n$ that bid prices $x_1, \ldots, x_n$ on an item. Each bidder has a key k_i and sends encrypted

$$E_{k_1}(x_1), \ldots, E_{k_n}(x_n) \to \text{Auctioneer}$$

Suppose that $x_1 > \cdots > x_n$. Then at the end of the auction, B_1 gets the item and pays x_2. The bidders then want a ZKP of validity of the auction, without revealing bid prices.

7.3 Regulation/compliance

ZKPs of compliance [Hal Varian].

Scenario: there are hospitals $H_1, \ldots, H_k$ and interns $I_1, \ldots, I_m$. Interns have list of hospital they want to go to. Hospitals then match up/rank interns, and then have a proof of correctness without divulging the actual rankings.

7.4 Multi-party computations

There are computing entities $P_1, \ldots, P_{15}$. Secrets x, y are distributed as shares $x_1, \ldots, x_{15}$, $y \mapsto y_1, \ldots, y_{15}$ to the entities. As long as at most 5 entities are improper, multi-party computation will compute whether $x > y$ while holding complete secrecy (no one finds out the values of x, y).

7.5 Homomorphic encryptions

Paillier encryption is a homomorphic encryption. For $n \sim 2^{2000}$, the encryption $E_n()$ satisfies

$$E_n(x_1, r_1) + E_n(x_2, r_2) \bmod n^2 = E_n((x_1 + x_2) \bmod n, (r_1 r_2) \bmod n)$$

Computations are $\bmod 2^{2000}$, which is a lot of bits.

8 Secret Sharing

8.1 Classical secret sharing [A. Shamir]

8.1.1 Polynomial interpolation

Let E be any field.

Theorem 8.1. *Suppose we have $(x_1, y_1), \ldots, (x_k, y_k) \in E \times E$ with $i \neq j \implies x_i \neq x_j$. There exists a unique polynomial*

$$f(t) = a_0 + a_1 t + \cdots + a_{k-1} t^{k-1} \in E[t]$$

such that $f(x_i) = y_i$ for all $i \in [1, k]$.

Proof 1.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{k-1} \\ & & & & \\ 1 & x_k & x_k^2 & \ldots & x_k^{k-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ \\ y_k \end{bmatrix}$$

The above matrix is known as the Vandermonde matrix, and it is well known that it has determinant $\prod_{i<j}(x_i - x_j) \neq 0$. Thus the system of equations has a unique solution $a_0, \ldots, a_{k-1}$. □

There is a $k^2 \log k$ computation of the inverse of the Vandermonde matrix.

Proof 2 (Lagrange Interpolation). Define the Lagrange polynomials

$$g_i(t) = \frac{(t - x_1) \cdots \widehat{(t - x_i)} \cdots (t - x_k)}{(x_i - x_1) \cdots \widehat{(x_i - x_i)} \cdots (x_i - x_k)}.$$

We have that $g_i(x_j) = \delta_{ij}$. Now just use y_i as the scalars.

Uniqueness: assume $f_1(t), f_2(t)$ both have degree $k - 1$. Then $f_1(t) - f_2(t)$ has roots at x_i. There are k roots $x_1, \ldots, x_k$, but the degree is $k - 1$, so the difference must be identically 0. □

8.1.2 (n,k) secret sharing algorithm

Let there be n persons $P_1, \ldots, P_n$ and a secret s. An (n,k) *secret sharing* is a protocol where each P_i gets a share s_i of s so that

1. Any k shares completely determine s.
2. Any $k-1$ (or fewer shares) reveal no information about s.

A. Shamir proposed the following secret sharing algorithm. Dealer takes $s \in \mathbb{F}_p$ and $n < p$. Assign each P_i a distinct $0 \neq \alpha_i \in \mathbb{F}_p$. Dealer chooses $a_1, \ldots, a_{k-1} \in \mathbb{F}_p$ randomly. Defines

$$f(t) = s + a_1 t + \cdots + a_{k-1} t^{k-1}.$$

Let $s_i = (\alpha_i, f(\alpha_i))$.

Theorem 8.2. *The above is an (n,k) secret sharing of s.*

Proof. (1) is obvious by polynomial interpolation. (2) Let $s_1, \ldots, s_{k-1}$ be given. Consider

$$(0,r), (\alpha_1, s_1), \ldots, (\alpha_{k-1}, s_{k-1})$$

where $r \in \mathbb{F}_p$ is completely random. Now we can use polynomial interpolation to get $h(t)$ such that h has constant term r and $h(\alpha_i) = s_i$. Thus no information is revealed. □

10/22 Some of the shares being given may be false shares. Solution: the dealer gives signed shares $(s_i, SIG_D(s_i))$ to P_i. Now the shares may be verified when pooled together.

8.2 Check vectors

Suppose we have a dealer D with some secret $s \in \mathbb{F}_p$. He sends it to intermediary I who will later pass it to a recipient R. The recipient wants to verify that he is receiving the same secret that I was given. The motivation for this is that D may have sold the rights to some product to I, who is now distributing to R.

D chooses $a, b \in F_p$ randomly and calculates $r = as + b \in \mathbb{F}_p$. He sends a, b to R at the beginning, and sends s, r to I. Then when I wants to pass on the secret, he sends s, r to R who can check that the relation $as + b = r$ holds. The probability that, given no other knowledge, I can [randomly] generate a pair r', s' such that $r' = as' + b$ is $1/p$.

Check vectors are further explored in PS 5, Problem 4.

8.3 Verifiable secret sharing

What if the dealer D is dishonest and doesn't send the correct secrets?

Suppose that $\mathbb{F}_p$, G a cyclic group, $|G| = p$, and generator $g \in G$ are all public. Assume g^x reveals nothing about x, i.e., intractability of discrete log. Dealer D wants to (n,k)-share a random $s \in \mathbb{F}_p$ amongst $P_1, \ldots, P_n$ where $n < p$. The $P_1, \ldots, P_n$ can broadcast values. D takes random $a_1, \ldots, a_{k-1} \in \mathbb{F}_p$ and posts $c_0 := g^s, c_1 := g^{a_1}, \ldots, c_{k-1} := g^{a_{k-1}}$. The secret is $\log_g c_0 = s$.

$$f(t) = s + a_1 t + \cdots + a_{k-1} t^{k-1}$$

Let $P_1, \ldots, P_n$ correspond to coordinates $1, \ldots, n$, which is possible since $n < p$. Now let $s_i = f(i)$. The Dealer should send s_i to P_i.

We show that P_i can verify that his share from D is proper. P_i computes g^{s_i}. He also calculates and checks that

$$c_0 c_1^i c_2^{i^2} \cdots c_{k-1}^{i^{k-1}} = g^{s + a_1 i + \cdots + a_{k-1} i^{k-1}} = g^{f(i)} = g^{s_i}.$$

If true, this asserts that $f(i) = s_i$. If P_i is honest and verification does not succeed, he posts "My share is false."

Assume that there are $< k$ bad people.

Theorem 8.3. *If there were m postings of claims of falsity, and $n - m \geq 2k - 1$, then the secret s is reconstructible.*

Proof. There are $2k-1$ players, say $P_1, \ldots, P_{2k-1}$ who did not post claim of false share. There are at most $k-1$ bad players amongst the above $2k-1$. Therefore there are at least k good players who have proper shares and upon demand will reveal their shares. Since $\mathbb{F}_p \to G : x \mapsto g^x$ is a bijection, we will know which shares s_i are proper by the verification. To reconstruct, the people with proper shares will pool their shares. There will be at least k such shares, so $f(t)$ can be interpolated. □

Note that since g^s is publicly available, if people know $s \in [0, 35]$, they can brute force find s. To fix this, take secret $\bar{s} \in \{0,1\}^{100}$, where $p \sim 2^{200}$. Dealer takes random $r \in \{0,1\}^{100}$ and sets $s = r \wedge \bar{s}$ (concatenate).

8.4 Sharing multiple secrets

In this section, assume that $< k$ people are gossipy, but not malicious. Everyone else is good and follows protocol.

8.4.1 Linear combinations of secrets

10/27 Suppose there are dealers $D_1, \ldots, D_m$ with values $v_1, \ldots, v_m \in \mathbb{F}_p$. The coefficients $\lambda_1, \ldots, \lambda_m \in \mathbb{F}_p$ are publicly known. The dealers want to jointly create an (n,k) sharing of $\lambda_1 v_1 + \cdots + \lambda_m v_m$ amongst persons $P_1, \ldots, P_n$ that does not reveal v_i.

Each D_j creates (n,k) sharing of $\lambda_j v_j$ via

$$f_j(t) = \lambda_j v_j + c_1^j t + \cdots + c_{k-1}^j t^{k-1},$$

letting $f_j(\alpha_i)$ be P_i's share of $\lambda_j v_j$. Assume there are secure private communication links $P_i \leftrightarrow P_\ell$ and $D_j \leftrightarrow P_i$. D_j sends $f_j(\alpha_i) \to P_i$. P_i has m shares $f_1(\alpha_i), \ldots, f_m(\alpha_i)$. The share for $\lambda_1 v_1 + \cdots + \lambda_m v_m$ is then $f_1(\alpha_i) + \cdots + f_m(\alpha_i)$. Interpolating will give the linear combination.

Note that since there are $< k$ gossipy people, there will not be enough people to pool shares and reconstruct v_i, since that is a breach of protocol.

8.4.2 Products of secrets

Suppose that secrets a_0, b_0 have been (n,k) shared with $P_1, \ldots, P_n$ using degree $k-1$ polynomials f, g. Assume $2k \leq n$. Then there is a way to (n,k) share $a_0 b_0$ such that reconstruction does not need to reveal a_0 or b_0.

Observe that $\deg h(t) = \deg f(t)g(t) = 2k-2$ (we multiply together two $k-1$ degree polynomials with different secrets).

$$h(t) = a_0 b_0 + d_1 t + \cdots + d_{2k-2} t^{2k-2}.$$

Consider the following Vandermonde matrix A:

$$\begin{bmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{2k-2} \\ \vdots & & & \vdots \\ 1 & \alpha_{2k-1} & \ldots & \alpha_{2k-1}^{2k-2} \end{bmatrix} \begin{bmatrix} a_0 b_0 \\ d_1 \\ \vdots \\ d_{2k-2} \end{bmatrix} = \begin{bmatrix} h(\alpha_1) \\ \vdots \\ h(\alpha_{2k-1}) \end{bmatrix} = \begin{bmatrix} f(\alpha_1)g(\alpha_1) \\ \vdots \\ f(\alpha_{2k-1})g(\alpha_{2k-1}) \end{bmatrix}$$

The $\alpha_1, \ldots, \alpha_{2k-1}$ are public knowledge. Thus anyone can calculate A^{-1}. Let the first row of A^{-1} be $(\lambda_1, \lambda_2, \ldots, \lambda_{2k-1})$. Then

$$\lambda_1 h(\alpha_1) + \cdots + \lambda_{2k-1} h(\alpha_{2k-1}) = a_0 b_0.$$

Now if we let $m = 2k-1$ with dealers $D_1, \ldots, D_m$ being $P_1, \ldots, P_{2k-1}$, by Section 8.4.1, $a_0 b_0$ can be (n,k) shared amongst $P_1, \ldots, P_n$.

9 Time Lapse Cryptography using ElGamal

10/29
By Adrian

There is a trusted secure bulletin board. Every person P_i has a digital signature SIG_i, a private key s_i known only to P_i, and a public verification key v_i. Recall for Elgamal we have a cyclic group G with order p and an encryption key $h = g^x$, both of which are public. Encryption of M is done by choosing $y \in \mathbb{Z}_p^*$ randomly and then setting $E(M) = (g^y, h^y M)$.

Now the encryption: every P_i chooses $x_i \in \mathbb{Z}_p^*$ randomly and posts $SIG_i(g^{x_i})$ on the bulletin board. A naive strategy is to encrypt using ElGamal by taking $h = g^{x_1} \cdots g^{x_n}$ from each person. Decrypt by receiving x_i from each person. But this has a lot of problems – a person could refuse to share their part.

The true encryption: every person P_i will (n, k) share his part x_i of the key, producing polynomials

$$f_i(t) = x_i + a_1^i t + ... + a_{k-1}^i t^{k-1}$$

and setting $x_{ij} = f_i(j)$, which will be P_j's share of x_i. Then P_i securely sends $SIG_i(\text{id}, \text{Time}, i, j, x_{ij})$ to P_j. Also, P_i posts $SIG_i(\text{id}, \text{Time}, g^{x_i}, g^{a_1^i}, \ldots, g^{a_{k-1}^i})$ on the bulletin board, which enables share verification. If P_j wants to claim that the x_{ij} he got from P_i is not a share of x_i by $f_i(t)$, P_j will post $SIG_i(\text{id}, \text{Time}, i, j, x_{ij})$. Then using the share verification data, everyone can check whether the accusation is justified.

The TLCS consists of the bulletin board and the parties $P_1, \ldots, P_n$. Any outside user can then use this TLCS service to encrypt and then decrypt at a later time.

To encrypt, before time T any user goes to the bulletin board and chooses the g^{x_i}'s which are not justifiably accused of being incorrect. Suppose $I \subset \{1, ..., n\}$ contains the indices of the people who did not post wrong shares. Then the user sets $h = \prod_{i \in I} g^{x_i}$, chooses y randomly, and encrypts $E(M) = (g^y, h^y M)$.

To decrypt, we assume there are at least k honest parties. By assumption, they have correct shares of each of x_i for each $i \in I$. Then they can reconstruct x_i together for each $i \in I$, recreate h, and decrypt as in ElGamal.

In this presentation of TLC, we have assume a single trusted secure bulletin board. Things become more complicated when there are multiple bulletin boards, some of which may be improper. This issue is addressed by using a Byzantine Agreement protocol in Section 10.1.

10 Byzantine Agreement

11/18
By Adrian, with edits

Origins of the question: suppose there are n generals surrounding a city and one master commander. At least m generals must attack the city simultaneously in order to succeed. Some generals, and possibly the commander, are dishonest. Each general receives a private message from the commander telling him to attack or not. The generals want to talk amongst themselves such that they can guarantee that either all proper generals will attack simultaneously or nobody attacks.

Fact: if there are k improper generals, then it requires at least k rounds to reach a Byzantine Agreement. If $k < n/3$ they can reach agreement as follows: first round, each general sends their value to every other general; in the ith round, each general sends the values they received from everybody else in the $(i-1)$st round. Not great because exponential amounts of data exchanged.

Fact: more recent work has given an algorithm to do this with polynomial amounts of data.

Fact: it is impossible to reach complete Byzantine Agreement in the case of asynchronous messaging.

Fact: you can do it probabilistically with very little data. It even generalizes to the asynchronous case. We will do this today:

Suppose there are n generals $P_1, ..., P_n$ and fewer than k improper generals such that $n > 6k$. Each P_i has a message M_i. We assume the synchronous case, so there is a common clock which defines common rounds $(R_1, R_2, ...)$ which run $([0, T], [T, 2T], ...)$. Also assume that there is also a common public coin which is flipped and whose value is received simultaneously by each general. The value of the coin flip is revealed at time $mT - T/2$ in round m. (There are public services which generate public random numbers, but generating the common coin is also an interesting problem. It might seem that Byzantine Agreement (BA)

is needed to generate the common coin, but there are easier ways of doing it as well.) Also assume a default common message d.

Conditions:

0. Every proper player (PP) halts.
1. Every PP has the same message at end.
2. If all PP had $M_i = M$, then their final common value is M.

A *Faulty-Improper player (FP)* is a player who does not follow protocol.

Definition 10.1. An (n, k) *Byzantine Agreement Protocol* is one that if executed by n players, $\leq k$ of whom are faulty, then agreement (i.e., 0,1,2) is achieved.

We present an (n, k) BA protocol. The players know each other, and in particular know that there are n players total.

Each P_i has variables $m(i)$, $count(i)$, $temp(i)$, and $c(i)$. At start, set $m(i) = M_i$ (i.e. what he currently thinks is the message) $count(i) = 0$. Then at round m:

1. P_i sends $m(i)$ to all P_j's.
2. Receive $m(j)$ from others.
3. Set $temp(i)$ to be the plurality of received messages including his own (i.e. most common message). If there is a tie, choose one randomly.
4. Set $count(i)$ to be the number of times $temp(i)$ was received.
   Assume (1)-(4) were done in the first half of round m, i.e., $[(m-1)T, mT - T/2]$.
5. (Lottery) Set $c(i)$ to be the value of the common public coin (at time $mT - T/2$).
6. (Decision) If $count(i) > n/2$ and $c(i) = 0$ OR $count(i) \geq n - 2k$ and $c(i) = 1$, then set $m(i) = temp(i)$. ELSE set $m(i) = d$ (default common message).

Theorem 10.1. *Assume L rounds $R_1, \ldots, R_L$ of the BA protocol are run. If during execution of the (n, k) BA protocol, no more than $k < n/6$ players became faulty, then Byzantine agreement as per 0,1,2 is achieved with $Pr \geq 1 - 1/2^L$. Output value is $m(i)$ at end of R_L.*

There is also another protocol that gets BA in an expected number of rounds.

Notation: let $\mathcal{G}$ denote the set of proper players. Assume $n - k \leq |\mathcal{G}|$.

Lemma 10.2. *If $M_i = M$ for $> n - 2k$ proper players, then $m(i) = M$ for all later rounds for all proper players.*

Proof. The number of proper players $> n - 2k \geq n/2$. □

Lemma 10.3. *After performing one round (given arbitrary messages) the probability that all proper players end with a common message is at least $1/2$.*

Proof. Assume that one proper P_j has updated $count(j) \geq n - 2k$. Of these $n - 2k$ received equal values, at most k are from improper players. Then any other proper P_i will have at least $n - 2k - k > n/2$ messages agreeing with $temp(j)$. Hence all proper P_i will have $count(i) > n/2$, so their messages will agree if the common coin $c(i) = 0$ for that round in this case.

Otherwise, if all proper P_j have $count(j) < n - 2k$ after Step 4, then they will all choose the default message if the common coin $c(i) = 1$ for that round. □

Proof of Theorem 10.1. Once the proper players have agreed, by Lemma 10.2, they will continue to agree. Hence, after L rounds the proper players will not reach an agreement with probability at most $1/2^L$. This proves the theorem. □

10.1 Byzantine agreement: applications to TLC

Suppose there are parties $P_1, ..., P_n$ that want to execute the TLC service. Number of improper players $k < n/6$. There are $B_1, \ldots, B_t$ bulletin bounds at least one of which is proper. Then TLC is possible.

Recall that the objective of TLC is to create an encryption key g^x by time T_0, where x will be revaled at a later time $T_1 = T_0 + \delta$. Consider $\mathbb{F}_p$, a cyclic group G of order p, and a generator $g \in G$ (all public). Every P_i chooses $x_i \in \mathbb{F}_p$ randomly and $a_1^i, ..., a_{k-1}^i \in \mathbb{F}_p$. Set

$$f_i(t) = x_i + a_1^i t^1 + \cdots + a_{k-1}^i t^{k-1}.$$

The j-th share of x_i is $x_{ij} = f_i(j)$. Signed $SIG_i(x_{ij})$ sent to P_j. Then P_i creates a signed

$$Ver(i) = SIG_i(g^{x_i}, g^{a_1^i}, \ldots, g^{a_{k-1}^i})$$

and also sends to every P_j.

Phase I: For $Ver(i)$, $P_1, \ldots, P_n$ run BA on having the same $Ver(i)$. This is done in parallel for all i (run n BA in parallel). The result is for every $P_i \in \mathcal{G}$ all $P_j \in \mathcal{G}$ will have the same final value $Ver(i)$. For some improper party (IP) P_ℓ, the $P_j \in \mathcal{G}$ will end with "faulty P_ℓ". For other IP P_ℓ, the proper $P_j \in \mathcal{G}$ will end with agreement on $Ver(\ell)$.

Phase II: Every P_i who has objection to some $x_{\ell,i}$ sends $x_{\ell,i}$ to all. Then the players do BA. At the end of Phase II, some P_ℓ's were disqualified by all $P_j \in \mathcal{G}$. No $P_i \in \mathcal{G}$ is disqualified by proper players.

Question is: which g^{x_i} will be used in the making of encryption key? Some bad guys might give proper $Ver(i)$ but then walk away when its time to decrypt.

Phase III (uses the clause $n - 2k \leq count(i)$ in Decision): Currently parties $\mathcal{G}$ agree for every surviving P_j. $Yes(P_j) =$ I have received share from P_j. $No(P_j) =$ I have not received share from P_j. The parties run BA on $Yes/No(P_j)$. If fewer than k proper parties $P_i \in \mathcal{G}$ have received a share from P_j, then every $P_i \in \mathcal{G}$ receives in round 1 at least $n - 2k = (n - k) - k$ (recall $|\mathcal{G}| \geq n - k$) $No(P_j)$ messages from $P_1, \ldots, P_n$. At the end of Phase III, all $P_i \in \mathcal{G}$ have agreed on $\mathcal{Q}$ qualified players. If $P_j \in \mathcal{Q}$, then all of his information is in the hands of the $\mathcal{G}$ proper, and these proper players $\mathcal{G}$ have at least k shares x_j.

End Game: Every $P_i \in \mathcal{G}$ posts on all BB's $SIG_i(P_j, Ver(j))$ for all agreed qualified players in $\mathcal{Q}$. This finished at time T_0. A user U needing an encryption scans all BB's and chooses all g^{x_j} that received $\geq n - k$ postings $SIG_i(P_j, Ver(j))$ on any one BB. Since the postings on BB's are signed, even improper BB's cannot forge signatures of postings. Therefore if $\geq n - k$ postings $SIG_i(P_j, Ver(j))$ do appear, they must also appear on the proper BB. Since $\leq k$ parties are improper and the proper parties have all reached agreement through private communication, the postings exactly correspond to the $P_j \in \mathcal{Q}$. The user's public key is

$$g^x = \prod_{P_j \in \mathcal{Q}} g^{x_j}.$$

Last Remark: BA stopping in expected 3–4 rounds. At the end of each round with $Pr \geq 1/2$ all $P_i \in \mathcal{G}$ are in agreement. In the next round every $P_i \in \mathcal{G}$ will have $n - k \leq count(i)$. P_i knows all $P_j \in \mathcal{G}$. All $P_i \in \mathcal{G}$ have agreed that $count(j) \geq n - 2k$. Hence settle on $temp(j)$. P_i stops and sends message P_i-stopped to all. [??????]

11 Commitment Methods

Suppose that A, B are involved in a protocol. Player A has a value v and sends COM(v) to B to commit to v. Later A will send v to B to decommit, with the following properties:

1. Before A decommits, B has no information on v.
2. (Binding) A cannot decommit using $v' \neq v$.

One method: Use hash function H. Choose $r \in \{0,1\}^{100}$ randomly and then $\text{COM}(v) = H(v||r)$. This is not absolutely unbreakable, but with a good hash function it is computational difficult to find a second value to decommit.

Another method: Given group G of order p and two generators $1 \neq g_1 \neq g_2 \in G$ public. Assume discrete log for G is intractable. If A wants to commit to $v \in \mathbb{Z}_p^*$, he chooses $r \in \mathbb{Z}_p^*$ randomly. Then compute $\text{COM}(v) = g_1^v g_2^r =: g_0 \in G$. To decommit, $A \xrightarrow{v,r} B$. By the midterm, for any v' there exists r' such that $g_1^{v'} g_2^{r'} = g_0$. Therefore (1) is satisfied. If A can decommit using $v' \neq v$, then $g_1^v g_2^r = g_1^{v'} g_2^{r'}$ and A can compute $\log_{g_2} g_1 = \frac{r'-r}{v-v'}$. Discrete log is intractable, so we must have $v' = v$. Thus $COM()$ is binding.

12 Straight Line Computations

12.1 Introduction

Notes in this section are taken from Prof. Rabin's lecture slides. Sections 12.1.1 and 12.1.2 describe the problems we would like to develop technology to solve.

12.1.1 Secure, secrecy-preserving auctions

A Vickrey auction for an item involves n bidders $B_1, \ldots, B_n$ bidding values $x_1, \ldots, x_n$. If $x_1 > x_2, x_2 \geq x_3, \ldots, x_n$ then B_1 gets the item and pays x_2.

We have an Auctioneer called an Evaluator-Prover (EP) to whom $B_1, \ldots, B_n$ submit their bids in a secure way. The EP computes the winner, the price he pays, and posts a secrecy-preserving proof of correctness. Bidders want to know correctness of the announced result but wish to keep their bids secret.

To this end we need to support evaluation and proofs of correctness for predicates such as $x_i > x_j$.

12.1.2 Matching Problems [H. Varian]

Entities: $E_1, \ldots, E_k$; candidates: $C_1, \ldots, C_m$. E_1 has a preference list $C^1_{i_1}, \ldots, C^1_{i_{m_1}}$. C_1 also has a preference list $E^1_{j_1}, \ldots, E^1_{j_{k_1}}$. Preference lists are secret. EP computes stable matching and can ZKP correctness.

12.1.3 Existing Technologies

Varieties of ZKP and arguments:

- Proving $x \in L$ an NP language
- Proving circuit satisfiability (at the bit level)
- Using homomorphic encryption to prove statements about encrypted values
- The method of obfuscated circuits [A. Yao]
- Multiparty computations, hiding inputs, intermediate result

12.1.4 Rabin's solution

Suppose we have values $x \in \mathbb{Z}_p$ for a prime $p \sim 2^{32}$. We introduce a random representation $RR(x) = X = (u, v) \in \mathbb{Z}_p \times \mathbb{Z}_p$, which has value $val(X) = (u + v) \equiv x \bmod p$. To make a random representation, simply take random $u \in \mathbb{Z}_p$ and set $v = (x - u) \bmod p$. To commit to a random representation, use $COM(X) = (E_{k_1}(u), E_{k_2}(v))$. Auctioneer wants to prove statements such as $val(X) + val(Y) = val(Z)$ without revealing x, y, z.

Example of addition: $p = 17, x = 7, y = 7, x + y = z = 14$. $X = (3,4), Y = (15,9), Z = (8,6)$. Auc posts $(10, -10)$. Verifier takes random $c \in \{0,1\}$. If $c = 0$, then opens "envelopes" of first coordinate to see $3 + 15 = 8 + 10$. Never open envelopes of both first and second coordinate!

Definition 12.1. A *straight line computation (SLC)* is a sequence of values

$$x_1, \ldots, x_n, x_{n+1}, \ldots, x_N \tag{12.1}$$

in $\mathbb{F}_p$ where

1. $x_1, \ldots, x_n$ are inputs (e.g., bids submitted by $B_1, \ldots, B_n$).
2. $\forall m > n$, $\exists i, j < m$ such that $x_m \equiv (x_i + x_j) \bmod p$ or $x_m \equiv x_i x_j \bmod p$.
3. $x_N = f(x_1, \ldots, x_n)$ for some $f \in \mathbb{F}_p[x_1, \ldots, x_n]$ is the output of the SLC.

In a generalized SLC, we also allow

$$x_m = \begin{cases} 1 & \text{if } x_i < x_j \\ 0 & \text{if } x_i \geq x_j \end{cases}$$

where comparisons are done as numbers in $[0, p-1]$.

A *translation* of the SLC (12.1) is a sequence

$$X_1, \ldots, X_n, X_{n+1}, \ldots, X_N \tag{12.2}$$

for $X_i \in \mathbb{F}_p \times \mathbb{F}_p$ where

1. $X_1, \ldots, X_n$ are random representation of $x_1, x_2, \ldots, x_n$.
2. If $x_m = x_i + x_j$ in the SLC, then $X_m = X_i + X_j$.
3. If $x_m = x_i \cdot x_j$, then X_m is a random representation of $x_i x_j$.

12.2 The model

We assume that there are n parties $P_1, \ldots, P_n$ respectively holding input values $x_1, \ldots, x_n$. The parties wish to perform a SLC on the inputs and obtain the output $f(x_1, \ldots, x_n)$. They want this to be done in a secrecy preserving manner where nothing is revealed about the inputs and intermediate calculations except the final output value. At the same time the protocol must provide a ZKP of correctness of the output.

An Evaluator-Prover (EP) is the entity who receives the inputs $x_1, \ldots, x_n$ from the parties, outputs $f(x_1, \ldots, x_n)$ and provides a ZKP of correctness of the output value.

12.3 Translation of inputs

Each party P_i creates 45 random representations $X_1^i, \ldots, X_{45}^i$ of x_i. This is done using a random number generator RNG taking seed s_1^i. P_i sets $RNG(s_1^i, j) = v_j^i$, $u_j^i = x_i - v_j^i \bmod p$, and $X_j^i = (u_j^i, v_j^i)$ for $j \in [1, 45]$. For commitment, a second seed s_2^i is used to generate encryption keys. P_i sets $RNG(s_2^i, j) = k_j^i$ for $j \in [1, 90]$. Then

$$COM(X_j^i) = (E_{k_{2j-1}^i}(u_j^i), E_{k_{2j}^i}(v_j^i)), \quad j \in [1, 45].$$

P_i securely sends to EP s_1^1, s_2^1, x_1 and

$$SIG_{P_1}(COM(X_1^i), \ldots, COM(X_{45}^i)).$$

[2] For a thorough and detailed account of SLCs, refer to `http://isites.harvard.edu/fs/docs/icb.topic627920.files/SLC.pdf`. From here on, I will also abandon attempts to provide dates of lectures, since everything is in one big conglomerate.

EP checks that input from P_i is proper. To do this:

1. Generates $X_1^i, \ldots, X_{45}^i$ using seed s_1^i.
2. Generates $k_1^1, \ldots, k_{90}^1$ using seed s_2^i and also computes the encryptions $COM(X_j^i)$.
3. Verifies P_i's signature on $COM(X_1^i), \ldots, COM(X_{45}^i)$.

Next, EP posts $SIG_{P_i}(COM(X_1^i), \ldots, COM(X_{45}^i))$ for $i \in [1, n]$. The auction is closed.

Put the random representations of $x_1, \ldots, x_n$ from $P_1, \ldots, P_n$ into a matrix M.

$$M = \begin{bmatrix} X_1^1 & \ldots & X_{45}^1 \\ & & \\ X_1^n & \ldots & X_{45}^n \end{bmatrix}$$

Before ending the auction, the EP posted $SIG(COM(M))$. Since the commitments were signed by parties $P_1, \ldots, P_n$, the values of M cannot be changed by EP at a later time in the ZKP.

Using the seeds s_1^i, the EP extends M to give K additional random representations of each x_i. The matrix with all extra representations is

$$M_e = \begin{bmatrix} X_1^1 & \ldots & X_{45}^1 & X_{45+1}^1 & \ldots & X_{45+K}^1 \\ & & & & & \\ X_1^n & \ldots & X_{45}^n & X_{45+1}^n & \ldots & X_{45+K}^n \end{bmatrix}$$

Let M_e^+ consist of the columns of M_e not in M. Using s_2^i to generate new keys, EP posts $COM(M_e^+)$. The seeds are used for efficiency of storage.

We say that X_1^1 and X_2^1 are *pairwise value consistent (pvc)* if $val(X_1^1) = val(X_2^1)$. EP and Ver agree on $L := 20$ (any number in this vicinity also works). Conduct a ZKP using all 45 input columns (all of M).

Theorem 12.1. *If fewer than $45 - L$ columns of M are pvc or fewer than $(1 - \frac{2}{L})K$ of M_e^+ are pvc with the $45 - L$ pvc columns of M, then Pr(accepted under the ZKP)$\leq 1/10^8$.*

Proof. Ver randomly creates 45 $n \times (L + 1)$ matrices $M_{e,1}, \ldots, M_{e,45}$. As an example, we consider $M_{e,1}$. First column of $M_{e,1}$ is column 1 of M, and the other L columns randomly chosen from M_e^+. EP wants to ZKP to Ver that the columns of $M_{e,1}$ are pvc. To do this, for every pair, e.g., X_1^1, X_{46}^1, the EP gives Ver $z \in \mathbb{F}_p$ such that $X_1^1 = X_{46}^1 + (z, -z)$. Ver randomly chooses $c_1 \in \{1, 2\}$. The EP reveals the c_1 component of all representations in $M_{e,1}$. This is done for every $M_{e,i}$.

Lemma 12.2. *If fewer than $(1 - 2/L)K$ of the columns of M_e^+ are pvc, then Pr(accept)$\leq 1/10^8$.*

Proof. Consider column 1 of M and $M_{e,1}$. In the ZKP of EP and Ver, acceptance will happen if either (a) all columns of $M_{e,1}$ are pvc, or else (b) the random challenge $c_1 \in \{1, 2\}$ to open the c_1 component of each representation in $M_{e,1}$ does not catch inconsistency. (a) By assumption there are fewer than $(1 - 2/L)K$ columns of M_e^+ that have the same value as the first column of M. Therefore Pr(a occurs)$< (1 - 2/L)^L \approx 1/e^2$. Next if the columns of $M_{e,1}$ is not pvc, then passing the challenge depends on choice of c_1. Therefore Pr(b occurs)$\leq 1/2$. Therefore Pr(Ver accept ZKP for $M_{e,1}$)$\leq 1/2 + 1/e^2$. To justify summing the probabilities, we may either use heavy probabilistic analysis to show independence or argue that $K \gg L$ makes the difference insignificant. [Probabilistic analysis incomplete.] Hence Pr(Ver accepts $M_{e,1}, \ldots, M_{e,45}$)$\leq (1/2 + 1/e^2)^{45} \approx 1.37 \cdot 10^{-9}$. □

Lemma 12.3. *Assume that more than $(1 - 2/L)K$ of the columns of M_e^+ are pvc. If fewer than $45 - L$ of the original 45 columns of M are value consistent with the $(1 - 2/L)K$ majority of additional columns, then Pr(Ver accepting)$\lesssim 1/2^{20}$.*

Proof. Under the above assumptions there are at least L columns of M each not value consistent with the majority. Say the fifth column of M is not value consistent. Ver will accept ZKP of $M_{e,5}$ if either (a) all additional L columns of $M_{e,5}$ are taken from the $2/L \cdot K$ inconsistent with majority, or (b) $M_{e,5}$ is not pvc and random challenge $c_5 \in \{1,2\}$ fails to uncover. Pr(a occurs)$\leq (2/L)^L$. As usual, Pr(b occurs)$\leq 1/2$. Therefore Pr(Ver accepts all L inconsistent columns)$\leq (1/2 + (2/L)^L)^L \approx 1/2^L$. □

The conclusion is that Pr(Ver accepts M and M_e under the assumptions of the theorem) $\leq 1/2^L + 1/10^8$. □

Result: if Verifier accepts, then essentially of the untouched columns of M_e, a proportion of $1 - \frac{2}{L}$ are pvc (for $L = 20$, that's $9/10$, which is a lot). In the course of the proof one component of each column of M and $45L$ columns of M_e^+ are revealed. Thus the EP has $K - 45L$ columns that he can reveal in future ZKPs, of which Ver is convinced a fraction $1 - \frac{2}{L} = 9/10$ of are pvc with $45 - L = 25$ majority columns of M.

12.4 Building a translation of SLC; ZKP of correctness

The EP takes inputs $x_1, \ldots, x_n$ and computes the SLC as in (12.1) to get output $x_N = f(x_1, \ldots, x_n)$. Let $X_1, \ldots, X_n$ be a translation of $x_1, \ldots, x_n$. We want to extend this to a translation of the SLC (12.1) as in (12.2) in such a way that ZKPs of correctness are possible.

For m where $x_m = x_i + x_j$, we just let $X_m = X_i + X_j$.

Consider m where $x_m = x_i x_j$ for $i, j < m$ to the SLC. Let $X_i = (u_i, v_i)$ and $X_j = (u_j, v_j)$. The EP chooses random $r_m^2, r_m^3 \in \mathbb{F}_p$ and sets

$$X_m^1 = (u_i u_j, v_i v_j) \tag{12.3}$$

$$X_m^2 = (u_i v_j + r_m^2, -r_m^2) \tag{12.4}$$

$$X_m^3 = (u_j v_i + r_m^3, -r_m^3) \tag{12.5}$$

Define $X_m = X_m^1 + X_m^2 + X_m^3$. Now $val(X_m) = u_i u_j + v_i v_j + u_i v_j + v_i u_j = (u_i + v_i)(u_j + v_j) = x_i x_j = x_m$.

We will later see that there is a problem when x_i is both a multiplier ($x_m = x_i x_j$) and multiplicant ($x_m = x_j x_i$). To fix this, for every x_i which in the SLC is both multiplier and multiplicant, EP introduces in the translation an additional random representation RX_i of x_i. So the final translation looks like

$$X_1, \ldots, X_n, RX_1, \ldots, RX_n, \ldots, X_m, \ldots$$

For the EP to verify to Ver that this is indeed a translation of the SLC, the following aspects must be considered.

1. Ver chooses a random challenge $c \in \{1,2\}$. If $c = 1$, then all first coordinates of the translation are revealed, and Ver checks that *all* addition relations are true, $val(X_i) = val(RX_i)$, and for any $x_i x_j = x_m$, the first coordinate of X_m^1 is $u_i u_j$. Similarly if $c = 2$, then all second coordinates of the translation are revealed, and Ver checks that *all* addition relations are true, $val(X_i) = val(RX_i)$, and for $x_i x_j = x_m$, the second coordinate of X_m^1 is $v_i v_j$.

2. Check correctness of all relations (12.4). Ver asks for the first coordinate of X_i and the second coordinate of X_j. Ver also asks for both coordinates of X_m^2 and checks that $u_i v_j = val(X_m^2)$.
   Problem: what if x_i is both multiplier ($x_m = x_i x_j$) and multiplicant ($x_m = x_j x_i$) (e.g., $x_m = x_i^2$). This means you have revealed the value of x_i in the process of the proof.
   Solution: You check $x_i x_j$ using X_i, X_j, but when checking $x_k x_i$, use X_k, RX_i; in special case x_i^2, use X_i, RX_i.

3. Check correctness of all relations (12.5) as in Aspect 2.

If the translation is false wrt addition, $val(X_i) = val(RX_i)$, or relations (12.3) or (12.4), then Pr(Ver accepts)$\leq 1/2$. Also if the translation is false wrt Aspect 2 or Aspect 3, and Ver audits the same, then Pr(accept)$= 0$.

12.5 Full verification

In Full Verification, EP has posted M (45 columns) and M_e^+ (K columns). EP ZKPs value consistency to Ver. There will be $K - 45L$ columns of M_e left to use at the end, where $L = 20$. Ver is convinced that 9/10 of these are pvc.

Ver randomly selects 105 of the untouched columns of M_e^+. EP extends each column to full translation of the SLC as in Section 12.4. Let the extended SLC (ESLC) denote the collection of all 105 translations. If the output the EP gives is incorrect but Ver accepts the ZKP, that must mean that every translation in the ESLC is incorrect. For a translation to be incorrect, either (a) commitment to input values $x_1, \ldots, x_n$ is incorrect or (b) translation incorrect wrt Aspect 1, 2, or 3. Pr(a occurs)$= 1/10$. Out of all translations where (b) occurs, by pigeonhole more than $1/3$ of them are false wrt Aspect i for $i \in \{1, 2, 3\}$. Verifier chooses to audit Aspect 1 with Pr $1/2$, Aspect 2 with Pr $1/4$ and Aspect 3 with Pr $1/4$.

Assume more than $1/3$ false wrt Aspect 1. In this case, Pr(accepting)$\leq 1/2 + 1/2 \cdot 1/10 + 1/2 \cdot 9/10 \cdot 2/3 + 1/2 \cdot 9/10 \cdot 1/3 \cdot 1/2 = 0.925$. Now $(0.925)^{105} \leq 2.8 \cdot 10^{-4}$.

Assume more than $1/3$ false wrt Aspect 2. Then Pr(accepting)$\leq 3/4 + 1/4 \cdot 1/10 + 1/4 \cdot 9/10 \cdot 2/3 = 0.925$. Again, $(0.925)^{105} \leq 2.8 \cdot 10^{-4}$. The result is the same if more than $1/3$ are false wrt Aspect 3.

Thus we conclude that Pr(Ver accepts when output is wrong)$\leq 2.8 \cdot 10^{-4}$.

12.6 ZKP of inequalities

By Adrian

Remark: this section does not require the machinery of SLCs with addition and multiplication developed in previous sections.

Suppose $x, y < p/2$ and work in $\mathbb{F}_p$ such that $p \sim 2^{32}$. We want a ZKP that $x \geq y$. It is enough to prove that $x < p/2$, $y < p/2$, and $x - y < p/2$ which implies that $x \geq y$ since we are working $\mod p$ (otherwise the overflow will be greater than $p/2$). The EP generates random representations $X = (u_1, v_1)$, $Y = (u_2, v_2)$ such that $val(X) = x$, $val(Y) = y$.

A test set $TS = (Z_1, ..., Z_{40})$ is a collection of random representations such that $Z_j \in \mathbb{F}_p \times \mathbb{F}_p$ and $(val(Z_1), ..., val(Z_{40}))$ is a permutation of $(1, 2, 4, ..., 2^{19}, 0, ..., 0)$.

The EP/Auct wants to ZKP that $x \geq y$. Creates pairs of test sets $(TS_1^1, TS_2^1), (TS_1^2, TS_2^2), (TS_1^3, TS_2^3)$. EP posts $COM(X), COM(Y), COM(TS_1^i), COM(TS_2^i)$. Ver chooses $c_1 \in \{1, 2\}$ to send to EP. Suppose $c_1 = 1$. EP opens all envelopes of $COM(TS_1^i)$ for $i = 1, 2, 3$, and Ver checks that TS_1^i are actually test sets. Then $COM(X), COM(Y), COM(TS_2^i)$ are untouched, and the verifier knows with probability $1/2$ that $COM(TS_2^i)$ are valid as described above.

EP posts for TS_2^i the 20 nonzero indices $j_1^i, ..., j_{20}^i$ and $(r_i, -r_i)$ for each i such that

$$X = Z^i_{2,j_1^i} + \cdots + Z^i_{2,j_{20}^i} + (r_i, -r_i).$$

If the EP proves that the previous equation is true, then he has shown that $x < 2^{20}$ while revealing nothing about which bits are zero in the binary representation of x. Ver challenges $c_2 \in \{1, 2\}$. If $c_2 = 1$, then EP opens all the first coordinates of the Z^i_{2,j_k^i}.

In other words, EP chooses indices j such that

$$X = \sum_{k=1}^{20} Z^1_{2,j_k^1} + (r_1, -r_1)$$

$$Y = \sum_{k=1}^{20} Z^2_{2,j_k^2} + (r_2, -r_2)$$

$$X - Y = \sum_{k=1}^{20} Z^3_{2,j_k^3} + (r_3, -r_3)$$

This allows us to prove that $x, y < 2^{20}$ and $x - y < 2^{20}$. If one of these statements is false, then the verifier can find out with probability $1/2$ by opening half the envelopes in his challenge. If we repeat this 20 times, Pr(false but Ver accepts)$\leq 1/2^{20}$.

The End.