CS174 Spr 99 **Lecture 24 Summary** **John Canny**

Cryptography

The idea of cryptography is to protect data by transforming into a representation from which the original is hard to recover. These days many networking technologies (internet, wireless) allow many agents to see a piece of data as it moves from its source to its destination. Those agents can capture the data in the representation sent across the network, and use it for their own purposes. Increasingly, people send data of great value (e.g. credit card numbers, secrets) through networks, so there is plenty of need for crytpography. In the future, most monetary transactions will probably go across the standard internet, and cryptography is an essential part of doing those transactions safely. There are two common types of cryptography:

Public-Key systems In a public-key system, the message M is encrypted using a key e which is public. That is, to encrypt the message compute $X = E(M, e)$ where E is the encryption function. To decrypt the encrypted message X, you compute $M = D(X, d)$, where d is the decryption key corresponding to e. The point of public-key systems is that knowing the encryption key e doesnt help a spy to discover the decryption key d.

Private-Key systems In a private-key system, the message M is encrypted using a key e which is known only to the sender and receiver. Once again, to encrypt the message compute $X = E(M, e)$ where E is the encryption function. To decrypt X, you compute $M = D(X, d)$, where d is the decryption key corresponding to e. In a private-key system, there is usually a simple relationship between the encryption and decryption keys, so knowing e would make it easy for a spy to intercept and decrypt a message. Probably the most widely used secret-key system is DES (the Data Encryption Standard).

RSA: A public-key crypto-system

The most famous public-key system is called RSA after its inventors Ron Rivest, Adi Shamir, and Len Adleman. Its very easy to describe RSA given what we know about additive and multiplicative groups of $\mathbb{Z}_n$. First of all, we assume the message is broken into chunks of the right size, say 1024 bits. In what follows, assume M is at most 1024 bits.

1. Generate a number n of at least 1024 bits which is a product of two large primes p and q. i.e. generate two primes of at least 512 bits and multiply them together.
2. Given p and q, recall that $\phi(n) = (p-1)(q-1)$ so it is easy to compute $\phi(n)$.
3. For the encryption key e, choose a value s.t. $\gcd(e, \phi(n)) = 1$.
4. Using the extended Euclid algorithm, find the multiplicative inverse of $e (\mathrm{mod}\ \phi(n))$, that is, find x such that $ex + \phi(n)y = 1$. This inverse is the decryption key d.

5. The public (encryption) key is the pair (e, n), while the decription key, which only the receiver knows is (d, n).

To send a message using RSA, the sender computes

$$X = M^e (\text{mod } n)$$

And then to decrypt the message, the receiver computes:

$$M_0 = X^d (\text{mod } n) = M^{ed} (\text{mod } n)$$

Now since $M^{\phi(n)} (\text{mod } n) = 1$ and ed is 1 + some multiple of $\phi(n)$,

$$M^{ed} (\text{mod } n) = M^1 (\text{mod } n) = M$$

So the recovered message M_0 is indeed equal to the original message M. Notice that the encryption and decryption functions are identical, that is:

$$E(X, (k, n)) = D(X, (k, n)) = X^k (\text{mod } n)$$

We would like to be convinced that RSA is a secure cryptographic scheme. For it to be so, we need a few hypotheses:

Factoring is Hard It is believed to be very difficult to factor an integer n in the worst case. The worst case is where n comprises a small number of large factors. This has not been shown to be NP-complete, but the problem has resisted years of effort at finding efficient algorithms.

Discrete Log is Hard Given a and b, and a modulus n, the discrete log of b relative to a is the k such that $b = a^k (\text{mod } n)$. Finding k from a, b, n is believed to be extremely hard, although it has not been proved to be NP-complete.

To see how discrete log could be used to break RSA, suppose you the spy grab a person's public key (e, n) from a public directory of keys. You then pick a random message M and compute $X = M^e (\text{mod } n)$. If you had an efficient box that computed discrete logs, you could then ask for the log of M relative to X, that is, the k such that

$$M = X^k (\text{mod } n)$$

and as we have seen, this k value would be the decryption key d.

To see how factoring would help break RSA, notice that knowledge of the factors p and q of n is all that was needed in the key generation procedure described above. That is, given p and q, you can compute $\phi(n)$, and given $\phi(n)$ you can compute the decryption key from the encryption key using the extended Euclid algorithm.

A third weakness would be if we had an efficient way of finding $\phi(n)$ from n without factoring, because that would still allow us to recover d from e using extended Euclid. But in fact, this is equivalent to factoring. That is, knowledge of $\phi(n)$ allows us to factor n efficiently. This requires a

randomized algorithm for general n, but if we know n is a product of two primes, there is a simple deterministic procedure.

Lemma
Given n which is a product of two primes p and q, and $\phi(n)$, it is possible to recover p and q in polynomial time.

Proof
In this case $\phi(n) = (p-1)(q-1) = pq - p - q + 1$, so $(p+q) = n - \phi(n) + 1$. So we know the sum of p and q and we also know their product $pq = n$. That gives us two polynomial equations in two unknowns. We can solve them with standard algorithms, or by hand:

(A)$\ldots p + q - a = 0$

(B)$\ldots pq - b = 0$

and computing q(A)-(B) gives:

$q^2 - aq + b = 0$

which is a quadratic in one variable that is easily solved for q. Substituting into the sum or product equations gives p as well. QED

So if we know how to factor n or compute $\phi(n)$, then we can break RSA. But that only proves that these problems are at least as hard as RSA. We would like to do the opposite, that is, show that RSA is at least as hard as factoring. To do that, we have to show how to solve factoring problems using RSA. Then if RSA were easy, factoring would be easy too. But if factoring is hard as we suspect, RSA must be hard too. We first state this in terms of the keys. If a spy could figure out the decryption key given the encryption key, then they could factor n:

Lemma
Given n which is a product of two primes and d and e which satisfy $de = 1 (\mathrm{mod}\ \phi(n))$, there is a polynomial-time randomized algorithm for factoring n.

Proof
In this case, we dont know $\phi(n)$, but we do know d and e which gives us $m = (de - 1)$ which is a multiple of $\phi(n)$. This multiple will be enough to factor n.

Choose an $a \in \mathbb{Z}_n^*$ uniformly at random. We know that $a^m (\mathrm{mod}\ \ n) = 1$ because m is a multiple of $\phi(n)$. Now compute

$a_1 = a^{m/2} (\mathrm{mod}\ \ n)$

$a_2 = a^{m/4} (\mathrm{mod}\ \ n)$

$\ldots$

$a_t = a^{m/2^t} (\mathrm{mod}\ \ n)$

where t is the number of powers of 2 occuring in m. Let a_k be the first element in the sequence such that $a_k \neq 1$, (so $a_{k-1} = 1$), or a_t otherwise. Then compute $\gcd(a_k + 1, n)$. We claim that

with equal probability (1/4), this gcd is either 1, p, q, or n. That is, half the time, we separate out a factor of n. To see this, we use the isomorphim:

$$\mathbb{Z}_n \approx \mathbb{Z}_p \times \mathbb{Z}_q$$

which is implied by the chinese remainder theorem. Given any number mod n, we can uniquely represent it by its values $(\mathrm{mod}\ \ p)$ and $(\mathrm{mod}\ \ q)$. This is a particularly useful representation, because $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are fields. Now 1 in $\mathbb{Z}_n$ is represented as $1(\mathrm{mod}\ \ p)$ and $1(\mathrm{mod}\ \ q)$. Since $a_{k-1} = a^{m/2^{k-1}}(\mathrm{mod}\ \ n) = 1$, we have that

$$a^{m/2^{k-1}}(\mathrm{mod}\ \ p) = 1$$

$$a^{m/2^{k-1}}(\mathrm{mod}\ \ q) = 1$$

Now a_k is a square root of a_{k-1}, and in a field, the only square roots of 1 are ± 1. So

$$a^{m/2^{k}}(\mathrm{mod}\ \ p) = \pm 1$$

$$a^{m/2^{k}}(\mathrm{mod}\ \ q) = \pm 1$$

For a randomly chosen a, all these outcomes are equally likely. This is pretty easy to see. Each field $\mathbb{Z}_p$ is a cyclic group, and for a generator g, every element is a power of g. When we raise an element to the power $\phi(p)/2 = (p-1)/2$, the even powers of the generator will give +1 while the odd powers will give -1. Thus there are equal numbers of each. This argument holds even for an exponent which is an odd multiple of $(p-1)/2$.

Notice that $m/2^k$ will eventually be an odd multiple of $(p-1)/2$. That's because m is initially a multiple of $\phi(n)$ and hence a multiple of $p-1$. It is also a multiple of $(p-1)/2$ and when we remove enough 2's from its factorization, it will be an odd multiple. So for 1/2 of the choices of a its image $\mathrm{mod}\ \ p$ will be an odd power of a generator of $\mathbb{Z}_p$ and that will cause a -1 value to appear the first time $m/2^k$ equals an odd multiple of $(p-1)/2$.

A similar argument applies to a's image $\mathrm{mod}\ \ q$. There is a case analysis we should do here depending on whether $p-1$ and $q-1$ contain the same number of powers of two, but we will skip it to get to the important ideas. You should be able to fill it in yourself. We will do only the most complicated case where $p-1$ and $q-1$ contain the same number of twos. Then $m/2^k$ will be an odd multiple of both $(p-1)/2$ and $(q-1)/2$ at the same time.

Now consider $a_k + 1$. The value of $a_k(\mathrm{mod}\ \ p)$ is ± 1, so when we add one, we get either 0 or 2 $(\mathrm{mod}\ \ p)$. If it is zero, that means $a_k + 1$ is divisible by p, and there is a 50% chance of this. Similarly, there is a 50% chance that $a_k + 1(\mathrm{mod}\ \ q)$ is 0 or 2, and these probabilities are independent. Independence follows because of the isomorphism $\mathbb{Z}_n \approx \mathbb{Z}_p \times \mathbb{Z}_q$, so that sampling uniformly for a in $\mathbb{Z}_n^*$ corresponds to sampling uniformly and independently from $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and using chinese remaindering to get a.

So the four possibilities for $a_k + 1$ as residues $(\mathrm{mod}\ \ p, \mathrm{mod}\ \ q)$ are $(0,0)$, $(0,2)$, $(2,0)$, and $(2,2)$ and all are equally likely. In one-half the cases $(0,2)$ and $(2,0)$, $a_k + 1$ is divisible by only one of p or q, and so in those cases $\gcd(a_k + 1, n)$ will give a factorization of n. If we dont get a factorization, we can choose another a at random and repeat. QED