Recitation - Cryptography

1 A few reminders

- **Multiplicative set of integers modulo** N: $\mathbb{Z}_N^* = \{A \in \mathbb{Z}_N : \gcd(A, N) = 1\}$ (i.e. the set of elements in $\mathbb{Z}_N$ that have a multiplicative inverse.)
- **Totient function**: Euler's totient function, denoted $\phi(N)$, is the number of integers in the set $\mathbb{Z}_N$ that are relatively prime to N. $\phi(N) = |\mathbb{Z}_N^*|$.
- **Fast modular exponentiation**: To compute A^E mod N, repeatedly square A, always mod N. Multiply together the powers of A corresponding to the binary digits of E, again, always mod N.
- **Exponent universe:** If we are exponentiating an element $A \in \mathbb{Z}_N^*$, then we can think of the exponent as living in the modular universe $\mathbb{Z}_{\varphi(N)}$ (i.e. the exponent can be reduced modulo $\varphi(N)$).
- **Generator**: A generator G in $\mathbb{Z}_N^*$ is an element such that $\{G^0, G^1, G^2, \ldots, G^{\phi(N)-1}\} = \mathbb{Z}_N^*$. A generator is not guaranteed to exist in every universe $\mathbb{Z}_N^*$, however it is guarenteed to exist if N is a prime.
- **Computing multiplicative inverses in modular universe:** Suppose $A \in \mathbb{Z}_N^*$, i.e. it has a multiplicative inverse modulo N, i.e. $\gcd(A, N) = 1$. Then using Extended Euclid Algorithm, we can obtain k and ℓ such that $1 = kA + \ell N$. If we take this equation modulo N, we get that $kA \equiv_N 1$. Therefore k is the multiplicative inverse of A.

2 Diffie Hellman

Recall the Diffie-Hellman protocol for securely generating a secret key over a public communication channel:

Apoorva		Bhagwat
Picks a large prime P	(1)	
Picks a generator $B \in \mathbb{Z}_P^*$	(2)	
Randomly draws $E_1 \in \mathbb{Z}_{\phi(P)}$	(3)	
Computes $B^{E_1} \in \mathbb{Z}_P^*$	(4)	
Sends P, B, B^{E_1}	(5)	Receives P, B, B^{E_1}
	(6)	Randomly draws $E_2 \in \mathbb{Z}_{\phi(P)}$
	(7)	Computes $B^{E_2} \in \mathbb{Z}_P^*$
Receives B^{E_2}	(8)	Sends B^{E_2}
Computes $(B^{E_2})^{E_1} = B^{E_1 E_2} \in \mathbb{Z}_P^*$	(9)	Computes $(B^{E_1})^{E_2} = B^{E_1 E_2} \in \mathbb{Z}_P^*$

- In line 2, why must B be a generator?
- In lines 3 and 5, why are the random exponents chosen from the set $\mathbb{Z}_{\phi(P)}$?
- Lines 4, 6, and 9 involve modular exponentiation. How can we accomplish this efficiently?
- An eavesdropper can obtain $B, B^{E_1}, B^{E_2} \in \mathbb{Z}_P^*$. Can she efficiently recover $B^{E_1 E_2}$?
- Why is this protocol useful?

Solution.
- We want B^{E_1} to be a uniformly random element from the set $\mathbb{Z}_P^*$, not a smaller subset of $\mathbb{Z}_P^*$. For instance, it would be bad if B had order 2.
- Fermat's little theorem ($A^{P-1} \equiv 1 \pmod{P}$ when $A \nmid P$), so we can reduce the exponent modulo $P - 1$.
- Repeated squaring.
- There is no known efficient algorithm for finding E given B and $B^E \in \mathbb{Z}_P^*$ (discrete logarithm), so probably not.
- It can be used with one-time pad to form a private-key encryption system. Diffie Hellman key exchange is also the basis for the ElGamal public-key encryption system (see below) as well as the ElGamal commitment scheme.

■

3 ElGamal

Here a public-key encryption system that uses the Diffie-Hellman protocol for sharing a secret key. Suppose Apoorva wants to send a message M to Bhagwat.

Apoorva		Bhagwat
	(1)	Picks a large prime P
	(2)	Picks a generator $B \in \mathbb{Z}_P^*$
	(3)	Randomly draws $E_1 \in \mathbb{Z}_{\phi(P)}$
	(4)	Computes $B^{E_1} \in \mathbb{Z}_P^*$
Receives P, B, B^{E_1}	(5)	Sends P, B, B^{E_1}
Randomly draws $E_2 \in \mathbb{Z}_{\phi(P)}$	(6)	
Encodes M as an el. of $\mathbb{Z}_P^*$	(7)	
Computes $B^{E_2}, MB^{E_1E_2} \in \mathbb{Z}_P^*$	(8)	
Sends $(B^{E_2}, MB^{E_1E_2})$	(9)	Receives $(B^{E_2}, MB^{E_1E_2})$
	(10)	Computes $(B^{E_2})^{E_1} = B^{E_1E_2} \in \mathbb{Z}_P^*$
	(11)	Computes $(B^{E_1E_2})^{-1} \in \mathbb{Z}_P^*$
	(12)	Computes $(MB^{E_1E_2})(B^{E_1E_2})^{-1} = M \in \mathbb{Z}_P^*$

What is the public key? What is the private key?

Suppose $P = 17$, $B = 3$. Bhagwat sends Apoorva $(17, 3, 6)$ (line 5) (Note: $6 = 3^{15}$). Apoorva sends back $(7, 1)$ (line 9). What is the decrypted message?

Solution. Public key: P, B, B^{E_1}

Private key: E_1

7 is the decrypted message.

More specifically, $(B^{E_2})^{E_1} = 7^{15} \equiv 5 \pmod{17}$ by repeated squaring.
Then $5^{-1} \equiv 7 \pmod{17}$, so we finally compute $M = 1 \cdot 7 = 7$. ■

4 RSA

Receiver Protocol

1. Choose two large *distinct* primes P and Q
2. Compute $N = PQ$ and $\phi(N) = (P-1)(Q-1)$
3. Choose $E \in \mathbb{Z}_{\phi(N)}^*$
4. Publish the *public key*: (N, E)
5. Compute the *decyption/private key*: $D = E^{-1} \in \mathbb{Z}_{\phi(N)}^*$
6. Upon receipt of ciphertext C, compute $M = C^D \in \mathbb{Z}_N^*$

Sender Protocol

1. Encode M as an element of $\mathbb{Z}_N^*$
2. Send $M^E \in \mathbb{Z}_N^*$

Questions:

- (i) First, write down explicitly the values that the Receiver knows and the values an Adversary knows.

- (ii) What is the advantage that the Receiver has over the Adversary that allows the Receiver to recover the original messsage while it seems the Adversary cannot?
- (iii) Suppose the factoring problem can be solved in polynomial time (i.e. given some number, we can find in polynomial time its factors). How can you break RSA?
- (iv) Why must P and Q be distinct?
- (v) Why must the encrpytion key E be an element of $\mathbb{Z}^*_{\phi(N)}$?
- (vi) What if $M \in \mathbb{Z}_N \setminus \mathbb{Z}^*_N$? Is this something the receiver needs to worry about?

Solution.

- (i) Receiver: $C, P, Q, N, \varphi(N), E, E^{-1}$, where $C = M^E \in \mathbb{Z}^*_{\phi(N)}$, $N = PQ$, $\phi(N) = (P-1)(Q-1)$, $E \in \mathbb{Z}^*_{\phi(N)}$, $E^{-1} \in \mathbb{Z}^*_{\phi(N)}$.
  Adversary: C, N, E.
- (ii) The crucial advantage that the Receiver has is that she knows what $\varphi(N)$, and she knows what $\varphi(N)$ because she knows the P and the Q such that $N = PQ$. This allows her to compute $\varphi(N) = (P-1)(Q-1)$. And once she knows $\varphi(N)$, she knows what modular universe the exponent E lives in, which then allows her to compute $E^{-1} \in \mathbb{Z}^*_{\phi(N)}$ (using Extended Euclid Algorithm).

  The adversary knows N, but we believe he cannot, in polynomial time, compute $\varphi(N)$ from N. If he can compute $\varphi(N)$, then he can do exactly what the Receiver does and compute $E^{-1} \in \mathbb{Z}^*_{\phi(N)}$.
- (iii) If the factoring problem can be solved efficiently, then given N, the Adversary can compute P and Q such that $N = PQ$, and so can compute $\varphi(N)$ as the product $(P-1)(Q-1)$. And as described above, once he knows $\varphi(N)$, he can compute E^{-1}.
- (iv) If $P = Q$, then $N = P^2$. But determining if a number has a square root, and determining what the square root is, can be done efficiently. So we could break RSA.
- (v) If E is in $\mathbb{Z}^*_{\phi(N)}$, then it has a multiplicative inverse E^{-1} modulo $\varphi(N)$. This inverse is what allows the Receiver to recover the original message M since $C^{E^{-1}} = M^{EE^{-1}} = M$.
- (vi) If M is not a member of $\mathbb{Z}^*_N$, then it shares a factor with N. Performing Euclid's algorithm on M and N will reveal this, and then P and Q will be known to the sender, who knows M. This is extremely unlikely, however (there are $P + Q - 1$ many such M, which is small compared to PQ when P and Q are large).

■

5 ElCommit

A commitment scheme is a cryptographic primitive that allows one to commit to a chosen value while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes are designed so that a party cannot change the value after they have committed to it.

An application of this scheme is for someone to send data that can only be confirmed at a later time. For example, one can predict the results of some event occurrence, commit it to someone else, and then reveal the commitment once the event occurs. This should confirm that the prediction was valid and prevent the other party from knowing the contents of the prediction until the commitment is revealed.

A way to visualize a commitment scheme is to think of a sender (Alice) as putting a message in a locked box, and giving the box to a receiver (Bob). The message in the box is hidden from Bob, who cannot open the lock himself. The message inside the box cannot be changed, but merely revealed if Alice chooses to give Bob the key at some later time.

Let's see a digital implementation of this idea. Alice wants to commit to a value belonging to $\mathbb{Z}_P^*$, where P is a prime number. Recall that $\mathbb{Z}_P^*$ has a generator. Let's call the generator G. We assume P and G are publicly known. The commitment scheme has three high-level steps:

- (i) Alice commits to a value $M \in \mathbb{Z}_P^*$.
- (ii) Alice then later "opens" (or reveals) to Bob the value M.
- (iii) Bob verifies that M is indeed the value Alice has committed to originally.

How can we implement the 3 steps above so that the following 2 conditions are satisfied?

1. **Binding**: After step (i), Alice cannot lie and convince Bob that she committed to $M' \neq M$.
2. **Hiding**: After step (i), Bob does not learn anything about the message M.

Solution. https://www.youtube.com/embed/DZ9BQg_yeb0

In the first step above, Alice commits to a value $V \in \mathbb{Z}_P^*$ as follows: she picks $R, S \in \mathbb{Z}_{\varphi(P)}$ at random, and makes public the values $(A = G^R, B = G^S, V = M \cdot G^{RS})$ (the operations are done in the universe $\mathbb{Z}_P^*$).

In the second step Alice sends to Bob the values R and S.

In the third step, Bob first checks that R and S are correct by computing G^R and comparing it to A, and computing G^S and comparing it to B. Then he can compute $(G^R)^S = G^{RS}$. And finally he can compute $(G^{RS})^{-1}$ (how?), and use it to recover the committed value M.

Binding: Let $\log_G(X)$ denote the value Y such that $G^Y = X$. Note that once Alice sends $(A = G^R, B = G^S, V = M \cdot G^{RS})$, she is necessarily committing to the value

$$V \cdot (G^{\log_G(A)\log_G(B)})^{-1},$$

(which is denoted by M above).

The only way Alice can lie and convince Bob that she committed to a different value M' is by sending incorrect value of R or S. If, for instance, she sends R' instead of R, Bob can detect this because we cannot have $G^R = G^{R'}$. This follows from the fact that G is a generator and therefore the powers G^i are distinct for $i \in \{0, 1, \ldots, P-1\}$.

Hiding: We won't be completely formal here but informally, the *Decisional Diffie-Hellman (DDH) assumption* states that one learns nothing about G^{RS} given $A = G^R$ and $B = G^S$. In other words, given $A = G^R$ and $B = G^S$ for random R and S, Bob cannot tell the difference between G^{RS} and some uniformly random element X of $\mathbb{Z}_P^*$. So from Bob's perspective, V is $M \cdot X$ where X is a uniformly random element of $\mathbb{Z}_P^*$. This implies V is a uniformly random element of $\mathbb{Z}_P^*$ (why?), hiding all the information about M. ■