

# Final Practice



Copyright Aria Diamond 2021

# Some More Public Key Cryptography

# Classes of Attack with Public Key Ciphers

What do all these mean????????????????


- One wayness
- IND
- IND-KPA
- IND-CPA
- IND-CCA
- IND-key-leakage
- IND-key-dependent-plaintext

# Classes of Attack with Public Key Ciphers

What do all these mean????????????????

- One wayness:
- IND: an attacker can read passive traffic. They can encrypt messages (because the public key is public!)
- IND-KPA: this doesn't really exist because  $IND = IND-CPA$
- IND-CPA: an oracle encrypts messages now. This has the same security assumptions as IND.
- IND-CCA: an oracle can decrypt messages now (but not the challenge ciphertext).
- IND-key-leakage: an intrusion can reveal part/all of the key.
- IND-key-dependent-plaintext: the message contains the key bits. No one really uses this.

# RSA: a review

- Okay we have  $e$ ,  $d$  , and  $N$ . What do we do with them?
- How do you encrypt?
- How do you decrypt?

# RSA: a review

- Okay we have  $e$ ,  $d$  🦴, and  $N$ . What do we do with them?
  - Publish  $e$  and  $N$ , and keep  $d$  secret
- How do you encrypt?
  - $m^e \bmod N = c$
- How do you decrypt?
  - $c^d \bmod N = m' = m$

# RSA Variants

- Textbook: one-gayness.
- Padded RSA: is IND-CPA
- PKCS 1.5: can be attacked when there are short messages unless the randomness is at least  $1/2$  of the ciphertext.
- OAEP: (non-adaptive) IND-CCA secure. This is used in practice all the time
- OAEP+: adaptive IND-CCA secure. Much more expensive than OAEP, so it isn't used in practice.

# Whitman and Martin agree on Keys

Also known as Diffie Hellman or a Discrete Logarithm problem

- Grab a big prime and name it Cody (P). Grab a random number between 1 and Cody and call it Wyatt (G).
- Now say Gabi and Aria want to talk about their fan-fic. Gabi creates Cody and Wyatt, and then chooses another random value Viviana (x) (between 1 and Cody again), and sends Aria Cody, Wyatt, and  $Wyatt^{Viviana} \bmod Cody$  or  $G^x \bmod P$  for boring folks.
- Aria chooses another random value Valentina (her favorite yarn color) (y) (between 1 and Cody), and sends  $Wyatt^{Valentina} \bmod Cody$  or  $G^y \bmod P$  back to Gabi
- The shared secret is  $G^{xy} \bmod P = G^{yx} \bmod P$  which Gabi and Aria can compute because they have x or y, but an eavesdropper cannot compute because they don't.



# Diffie Hellman part 2

**Whitman and Martin are back! (Well we didn't leave them yet)**

- Oh hey! We do this for the Forks Method, in which we expand to arbitrary amounts of people. Why does that work?
- What's to stop Adam from saying he is Aria and exchanging keys with Gabi?
  - Nothing! And now he gets the first look instead of Aria with plausible deniability because he made a big show of him being above that fan-fic stuff.
- Okay then how do we fix it?

# Diffie Hellman part 2

- Why does the Forks Method work?
  - Groups have the associativity property and DH uses the group  $Zp^*$  with the operation of modular exponentiation, so we can swap the order of exponentiations and it still works!
- What's to stop Adam from saying he is Aria and exchanging keys with Gabi?
  - I mean there is the fact that Aria intimidates Adam, so ͸(ツ)͸
- Okay then how do we fix it?
  - Add authentication! Use a signature and PKI to securely exchange public keys for verification

# ElGamal

- An encryption algorithm based on Diffie Hellman (original DH only does key agreement).
- Gives IND-CPA with a ciphertext size 2x the plaintext (apparently this is good).
- Cramer Shoup: adaptive IND-CCA
- Cramer Shoup Lite: non-adaptive IND-CCA

# Reductions

**What problem is harder than the other?**

- For the factoring fun times, we have RSA? Is RSA or factoring harder or are they equally hard?
- Discrete log and decisional DH and computational DH?

# Reductions

**What problem is harder than the other?**

- For the factoring fun times, we have RSA? Is RSA or factoring harder or are they equally hard?
  - Factoring is a bit harder than RSA, so Factoring reduces to RSA.
- Discrete log and decisional DH and computational DH?
  - Decisional DH is the easiest, followed by computational DH, and then discrete log

# Homomorphic Encryption






# Definitions

What are these?

- Partially Homomorphic Encryption
- Malleable Encryption
- Fully Homomorphic Encryption

# Definitions

**Homomorphic Encryption:** encrypt data, send it to a server to process, get the result, decrypt

- Homophobic encryption: Yass Queen      Flannel
- Partially Homophobic Encryption: You can do some operations on encrypted data, but not all, and in specific orders, not arbitrary. This combines a bunch of cipher texts and creates a single output ciphertext from the operations.
- Malleable Encryption: operations are done on a specific ciphertext to transform it into another ciphertext.
- Fully Homophobic Encryption: You can do **any** operations (arbitrary boolean/arithmetic circuits) in **arbitrary** order. Still does the thing of operating on multiple cipher texts and returning a single ciphertext.



# Schemes we already studied!

Ronald pt 2 (and not even the last of him!)

- ElGamal: this is both product homomorphic ( $E(a) * E(b) = E(a * b)$ ) and product malleable ( $E(a) * j = E(a * j)$ )
- RSA: this is product homomorphic.
- What's fun is that this product homomorphism makes pure RSA insecure for signatures. If you have two signed messages  $m_1$  and  $m_2$  with  $s_1$  and  $s_2$  respectively, you can get  $m_1 * m_2$ , with valid forged signature  $s_1 * s_2$

# A bunch of new schemes! But first basics

They are all IND-CPA secure

- What is quadratic residuosity?
- What is isomorphism?
- What is an arithmetic/boolean circuit?

# A bunch of new schemes! But first basics

They are all IND-CPA secure

- What is quadratic residuosity?
  - For a random big value, does the square root exist, and if so, what is it?
- What is isomorphism?
  - This is graph theory stuff where can you transform one graph into another? So if you were to take the labels from one and map it to the other, would it be the same graph?
- What is an arithmetic/boolean circuit?
  - Boolean circuits are how computers work. This is like XOR, AND, NOR, etc.
  - Arithmetic circuits are plus and multiplication possibly modulo a value?

# Goldwasser Micali

- Based on quadratic residuosity. So is  $\sqrt{y}$  an integer? (Not what is the residue, just does it exist)
- This could only encrypt 1 bit at a time, so it is slow AF.
- This does xor homomorphism.

# (Probabilistic) Rabin

- This actually uses computing the square root of a value, not just whether it exists (much more efficient).
- This satisfies one-gayness, but an IND-CCA attack leaks the entire secret key!
- This is product homomorphic.
- Probabilistic Rabin: IND-CPA secure (thank ya). Converts the LSB to a hard-core predicate and does xor-malleability.

# Paillier

- Uses discrete logarithm math bs. Find the  $n^{\text{th}}$  root (rather than square root) of  $y$ . This also happens to be an isomorphism of  $Z_{n^2}^* = Z_n \cdot Z_n^*$
- This is sum homomorphic, sum (with a known value) malleable, and product (with a known value) malleable
- Sum homomorphism is useful for verifying votes publicly (provided you can keep certain keys in separate hands and not lose keys).

# Gentry

- The first fully homomorphic encryption algorithm (released in 2009)! This is too inefficient to use in practice.
- Has 2 steps
  - Somewhat homomorphism: a simple decryption to mask cipher texts with noise
  - Bootstrapping: re-encryption to lower noise.
- We can do addition (increments noise) and multiplication (doubles noise).

# DGHV

- An improvement of Gentry using public private key pairs (or just shared key too). This is more efficient.
- This uses boolean circuits instead of Gentry's arithmetic circuits.



# Signatures

# **Wait hold on one second....**

**What's a digital signature?**

- How is it similar and different from a MAC?

# Wait hold on one second....

## What's a digital signature?

- **Digital signature:** a way of validating that you and you alone sent a message. No one else could have sent the message, and no one could have modified the message.
- How is it similar and different from a MAC?
  - Well MAC's require shared keys, only people with the shared key can validate. Additionally, anyone with the shared key can create a MAC, but only you can sign.
  - However, MAC makes your face looks pretty (if we consider pretty to be following euro-centric beauty standards).

# Let's Sign things with Ronald and Friends

Ronald being Ron Rivest and the algorithm being RSA

- Okay but how?
- Okay but let's forge signatures!!
- Okay let's protect against forgery

# Let's Sign things with Ronald and Friends

Ronald being Ron Rivest and the algorithm being RSA

- Okay but how?
  - Decrypt the message and send it! You can verify by encrypting (because the public key is public) and should get the message.
- Okay but let's forge signatures!!
  - Encrypt a message! The message before you encrypted it is now a signature for the encrypted message.
- Okay let's protect against forgery
  - Hash the message and then decrypt! It's best if the hash space is all possible values in  $Z_n^*$

# ElGamal and Fiat-Shamir

They have a DSA child

- Schnorr: an identification scheme based on the hardness of discrete logarithm
- Fiat-Shamir: changes an identification scheme into an authentication scheme.
- ElGamal: does signatures based using discrete logarithms and hashing.

# Daddy Lamport!

**Daddy because he's a father of a bunch of things (Paxos, TLA+, Latex, precursor to vector clocks)  
He's like 70 so not that type of daddy!**

- You can sign with just a hash function! Pretty cool right? How?
- Well what is the fine print?

# Lamport Signatures (the fine print)

- You can sign a bit at a time with a hash. Reveal one of the two hashes based on the bit.
- You also cannot reuse the hashes otherwise you can break this.
- This just means that it's way too inefficient to use in practice. There are a couple of improvements (not really)
- Stateful: this sucks because it just generates a longer public key and secret key to match the length of the message
- Chain: this creates long chains, and the signature of a message reveals all the previous messages.



# Ralph is a treehugger

By Ralph we mean Ralph Merkle and he just created a tree based hash scheme (I'm the actual treehugger)

- This is similar or the same to Merkle hash trees used to efficiently verify the integrity of file systems (such as ZFS).
- A message is signed by using a key at the leaf.
- Each signature requires  $O(\log(t))$  operations.

# Elliptic Curve Cryptography

MATH! :(

- Cool things: there is no sub-exponential discrete logarithm algorithm for it, so you can have shorter keys and signatures!
- Not so cool things: the NSA backdoored the major curves that are now international standards!
- How it works:
  - Get a prime. Then get the roots by finding  $y^2 = x^3 + Ax + B \bmod p$ .
  - Now you can use this by doing  $x \cdot p$  instead of exponentiation.

# Public Key Infrastructure

**Like all good, corrupt infrastructure, it is managed by private companies**

- What is this?
- Why do we care? When do we use this?
- How does it work?

# Public Key Infrastructure

**Like all good, corrupt infrastructure, it is managed by private companies**

- What is this?
  - This is how you get the lock when you visit a website. It's the thing that does HTTPS and makes sure you are talking to the server you want to, not a spoofer.
- Why do we care? When do we use this?
  - You use this for everything. For banking, online shopping, classes, every website ever uses HTTPS and thus PKI.
- How does it work?
  - The server goes to a certification authority (CA) and gets a signed certificate saying it is a particular website. Clients can then verify the certificate is valid by tracing the signatures of on the certificate to a root authority installed on their computer by the OS.

# Hybrid Encryption

Zooooom!! 

- What is it?
- Why do we care?

# Hybrid Encryption

Zooooom!! 

- What is it?
  - Share keys with public key encryption, and then use the shared keys to do symmetric encryption for actual data transfer.
- Why do we care?
  - It gives us speed!! We need a way to share keys, and public key cryptography does that without having to send a private courier, and symmetric encryption is fast af.