Finished with Crypto Overview

Moving into Hashes and Public Key

- Scenarios:
  - Private key crypto
  - Public key crypto
• Overview of asymmetric-key crypto
• Intuition for El Gamal and RSA
  • And intuition for attacks
• Digital signatures / authenticity
• Recall our three goals:
  • Confidentiality
  • Integrity
  • Authenticity
We will use asymmetric crypto to mitigate these drawbacks!
High-level idea

- Generate a pair of keys
  - One for encryption, one for decryption
- Make encryption key public!
  - On your website, in the New York Times
  - Anyone can send you a private message
- Secret key is the *trapdoor*
Asymmetric crypto

Bob

Alice

Public channel

Em (or error)

• $k_e \neq k_d$
• $k_d =$ private key, $k_e =$ public key
  • Bob computes both, gives public key to Alice
• Alice sends a message to Bob: $c = E(m, k_e)$
• Bob can decrypt it: $m = D(m, k_d)$
• Anyone can send, only Bob can read!
Asymm. Cryptosystem: Definition

- Three polynomial-time algorithms:
  - KeyGen: Returns $k_p$ (public) and $k_s$ (secret)
  - $E(k_p, m)$: Encrypts $m$ with $k_p$, returns $c$ in $C$
    - Must be randomized (why?)
  - $D(k_s, c)$: Decrypts $c$ with $k_s$, returns $m$ in $M$
    - Or error

- Correctness condition:
  - For all pairs $(k_p, k_s)$: $D(k_s, E(k_p, m)) = m$
Security game

Challenge:
Choose $b = x$ or $y$ at uniform random

$$c = E(k_p, m_b)$$

$m_x$ and $m_y$

Eve’s job: Guess whether $x$ or $y$ was picked. Cipher text secure IFF no better than guessing
Public-key security

• Ciphertext-only security implies CPA security!
  • Why?
  • (Also implies multiple messages are OK)
Pros and Cons

• Scales well — everyone makes one key pair
  • Not \( n \) keys

• No direct setup comms between Alice and Bob

• Asymmetric is \textit{much, much slower}

• Asymmetric is easier to attack
  • Requires stronger assumptions
The authenticity problem

- In symmetric, we needed an **authentic, private** channel to exchange keys
  - Diffie-Hellman let us relax to **authentic** only
  - Public-key also requires authentic channel
- Who posted that ad in the NY Times?
  - Much more on this later
In practice: Hybrid

- Bob generates key pair and publishes $k_p$
- Alice generates new symmetric key $k_{AB}$
- Alice -> Bob: $c_1 = E(k_p, (\text{Alice} \ || \ k_{AB}))$
- Alice -> Bob: $c_2 = E(k_{AB}, \text{message})$
- Arbitrary-length messages, efficiently
  - Keep $k_{AB}$ as a session key
Intuition for algorithms
El Gamal (simplified)

- Similar to Diffie-Hellman
  - Public key: prime $p$, generator $g$, $h = g^x$
  - Private key: $x$

- Encryption: Sender chooses $y$
  - $c_1 = g^y$, $c_2 = m^y h^y$

- Decryption: $m = c_2 / c_1^x$

- Security equivalent to D-H hardness
RSA background

- N = pq, p and q distinct, odd primes
- $\phi(N) = (p-1)(q-1)$
  - Easy to compute $\phi(N)$ given the factorization of N
  - Hard to compute $\phi(N)$ without the factorization of N
- Fact: for all $x \in \mathbb{Z}_N^*$, it holds that $x^{\phi(N)} = 1 \mod N$
  - Proof: take CMSC 456!
- If $ed = 1 \mod \phi(N)$, then for all $x$ it holds that $(x^e)^d \mod \phi(N) = x \mod N$
  - I.e., given d, we can compute $e^{th}$ roots
Stated Another Way

• Public key is $e, N \quad c = p^e \mod N$
• Private key is $d, N \quad p = c^d \mod N$
• $d$ related to $e$ as mult inverse mod $\phi(N) = (p-1)(q-1)$
  • $d \times e \mod \phi(N) = 1$
  • mult inverse mod constant is easy (Euclid’s alg)
  • so $d$ can be found if we know $\phi(N)$

• How can $\phi(N)$ be found, assuming we know $n$?
  • factor into $p, q$, compute $(p-1) \times (q-1)$
  • but factoring big ints is hard
Hardness of computing $e^{th}$ roots?

- If factoring is easy, then the RSA problem is easy

- We know of no other way to solve the RSA problem besides factoring $N$
  - But we do not know how to prove that the RSA problem is as hard as factoring

- The upshot: we believe factoring is hard, and we believe the RSA problem is hard
How hard is factoring?

• Current record
  • factoring 768-bit number, collaboration of several big organizations.
  • Non-math dude w/ a couple computers can factor 512-bit numbers in a couple months

• So need $|N| \approx 1024$ for reasonable security

• Currently $|N| \approx 2048$ recommended for good security margins
RSA key generation

- Generate random primes \( p, q \) of sufficient length
- Compute \( N = pq \) and \( \phi(N) = (p-1)(q-1) \)
- Compute \( e \) and \( d \) such that \( ed = 1 \mod \phi(N) \)
  - \( e \) must be relatively prime to \( \phi(N) \)
  - Typical choice: \( e = 3 \); other choices possible
- Public key = \( (N, e) \); private key = \( (N, d) \)
“Textbook RSA” encryption

• Public key \((N, e)\); private key \((N, d)\)
• To encrypt a message \(m \in \mathbb{Z}_N^*\), compute
  \[ c = m^e \mod N \]
• To decrypt a ciphertext \(c\), compute \(m = c^d \mod N\)
• Correctness clearly holds…

• …what about security?
Textbook RSA is insecure!

- It is deterministic!
- Furthermore, it can be shown that the ciphertext leaks specific information about the plaintext
Padded RSA

• Introduce randomization…
• Public key (N, e); private key (N, d)
  • Say |N| = 1024 bits
• To encrypt \( m \in \{0, 1\}^{895} \),
  • Choose random \( r \in \{0, 1\}^{128} \)
  • Compute \( c = (r | m)^e \mod N \)
• Decryption done in the natural way…
• Essentially this is standardized as PKCS #1 v1.5 (since superseded)
Implementation attacks

• Timing and power:
  • How long / how much to compute $c^d \mod N$

• Bad randomness:
  • $p$ and $q$ can’t be predictably generated
  • If $n = pq$ and $n' = pq'$, both are broken

• Bad padding / malleability
Malleability

- Given $c$ (m unknown), can construct $c'$ that will decrypt to a related message $m'$
  - Recall CBC attack last time
- Basic El Gamal and basic RSA are malleable
  - CCA-safe variations exist
Activity
(Time permitting)
Public key example map
Message = 66
Private key map

Minimum dominating set = NP hard
Message = 66
Notes on this example

• Finding the (a) private map is very hard
  • Minimum dominating set (NP)
  • For a sufficiently large map

• But, can solve as a system of linear equations

• So, this is \textit{not secure}
  • But it is kind of a fun illustration
Zero-Knowledge Proof Systems

• Allows you to prove that you know a secret, w/o giving anything away. Is RSA one?

• Classic example: graph isomorphism
  • two graphs are isomorphic if we can rename the vertices of one and get the other

• How it works:
  • Alice creates $G_x$ a large (~500 vertices) graph (edges and vertices).
  • Alice creates $G_y$ by renaming vertices
  • Her secret key is the mapping of vertices between $G_x$ and $G_y$

• To prove that she is Alice, she:
  • Alice creates a series of graphs: $G_1$-$G_k$, each by renaming from $G_x$, and makes them public
  • For each $G_i$, Bob asks Alice to give him a mapping from $G_i$ to either $G_x$ or $G_y$, but not both. Bob chooses.
Does This Work?

- How can she do it?
  - renaming is trivial
- How is this zero-knowledge?
  - After the proof, Bob knows some graphs with mappings to $G_x$, and some with mappings to $G_y$.
  - However, he could have generated similar graphs himself.
  - He still does not know of any graphs w/ mappings to both $G_x$ and $G_y$.
- How is this proof?
  - Trudy could generate a series of graphs $G_1$-$G_k$ as well, however
  - She can only produce a mapping from a given $G_i$ to one of $G_x$ and $G_y$.
  - If Bob asks for the mapping she has, good. If not, she is exposed.
  - For 30 graphs, her chances would be 1 in $2^{30}$ (1 in a billion)
- Down side: too expensive.
Your turn! Public map
private map