Public Key Primitives

Rohit Musti

CUNY - Hunter College

March 23, 2022

Table of Contents

1. Overview
2. Trapdoor Functions
3. RSA
4. Diffie-Hellman Key Exchange

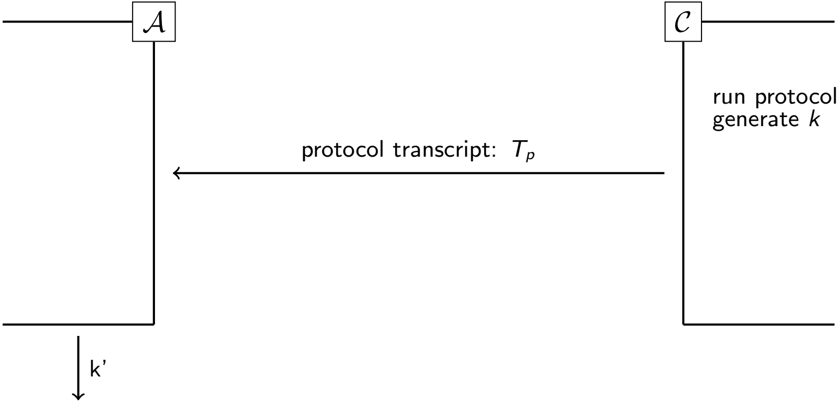
1 Overview

Public Key Exchange Motivation

- Consider our protagonists: Alice and Bob. They have never met in person and are speaking over the phone to coordinate a blind date!
- They want to make sure their date location is secret from any eavesdroppers listening to their phone line.
- They took introduction to cryptography and decide that they want to generate a shared secret c unknown to any adversary.
- This requires that if the eavesdropper takes the transcript of their phone call, they are not able to generate the secret k

NOTE: no requirements for integrity (no protection from man in the middle) and the protocol is fully anonymous (no way to verify that Alice and Bob are talking to one another)

Anonymous Key Exchange Attack Game



if $k' = k$, then the adversary wins

Weaknesses in this Security Notion?

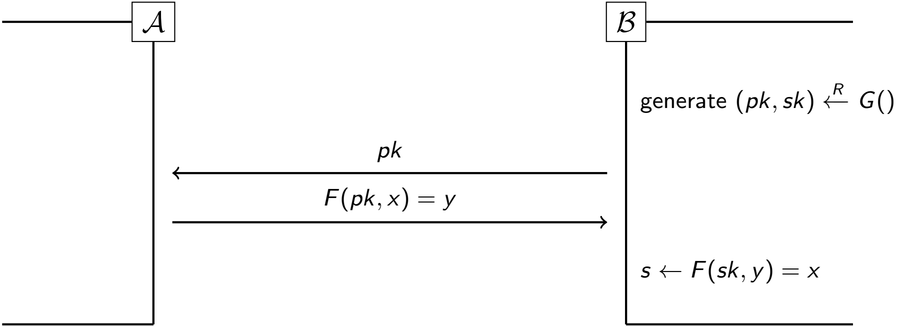
1. Assumes adversary will not tamper with protocol
2. Assumes that adversary cannot simply guess parts of k (i.e. no uniform randomness distinguishability requirement)
3. No identity verification

2 Trapdoor Functions

Trapdoor Functions

- Trapdoor functions are one way functions that have a "trapdoor" that allows someone armed with a secret to reverse the otherwise unreversible function
- Three functions over $(\mathcal{X}, \mathcal{Y})$: a generator, a function, and an inverter
  - G: probabilistic generator $(pk, sk) \overset{R}{\leftarrow} G()$
  - F: determinstic function $y \leftarrow F(pk, x)$
  - I: determinstic function $x \leftarrow I(sk, y)$ (should be hard w/o sk)
- correctness: $\forall (pk, sk) : I(sk, F(pk, x)) = x$

Trapdoor Key Exchange



$\mathcal{A}$

$\mathcal{B}$

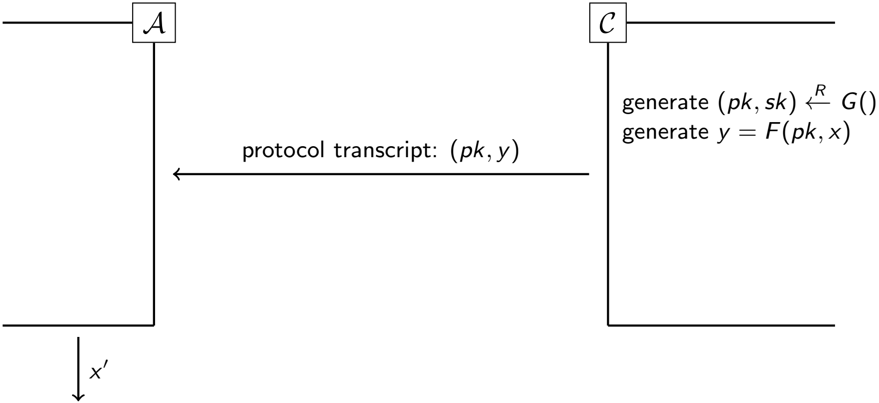
generate $(pk, sk) \xleftarrow{R} G()$

pk

$F(pk, x) = y$

$s \leftarrow F(sk, y) = x$

Trapdoor Key Exchange Attack Game



$\mathcal{A}$

$\mathcal{C}$

generate $(pk, sk) \overset{R}{\leftarrow} G()$

generate $y = F(pk, x)$

protocol transcript: (pk, y)

x'

if $x' = x$, then the adversary wins

3 RSA

RSA Background

- Named after Ron Rivest, Adi Shamir, and Leonard Adleman at the Massachusetts Institute of Technology
- Legend has it they got drunk on wine during passover at a student's house and came up with the system staying up all night
- allegedly, the british intelligence agencies came up with a similar system a few years earlier but didn't think it was feasible with the current computers

RSA Key Generation

- Key Generation
  1. pick an integer $\ell > 2$ and an odd integer $e > 2$
  2. generate a random ℓ-bit prime p s.t. $gcd(e, p-1) = 1$
  3. generate a random ℓ-bit prime q s.t. $gcd(e, q-1) = 1$ and $p \neq q$
  4. $n \leftarrow pq$
  5. $d \leftarrow e^{-1} mod(p-1)(q-1)$
  6. $pk = (n, e)$ and $sk = (n, d)$
- $x \in \mathbb{Z}_n$
- $F(pk, x) := x^e \in \mathbb{Z}_n$
- $I(sk, y) := y^d \in \mathbb{Z}_n$

RSA Security

- given n the RSA Modulus, e the encryption exponent, d the decryption exponent, and $y = x^e$, it is mathematically hard to calculate x

4 Diffie-Hellman Key Exchange

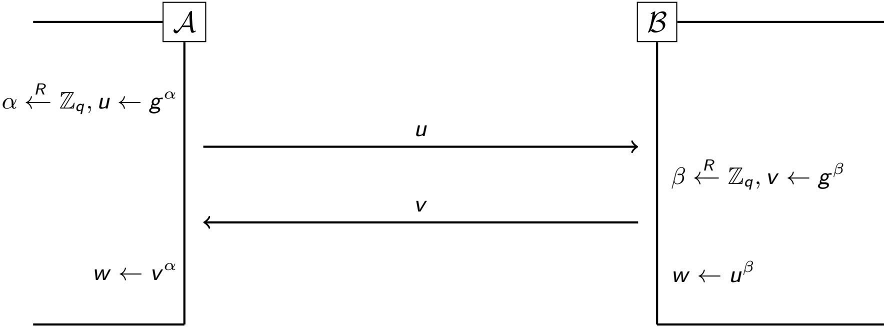
Diffie-Hellman History

- Earned the authors a Turing award
- Two Stanford Cryptographers Whitfield Diffie and Martin Hellman
- Before this time, little cryptography work was done outside of the NSA and other intelligence agencies
- NSA tried to limit their research after they published this public paper
- NSA even sent letters to journal editors warning that authors of cryptography papers could be sentenced to prison time for violating laws around military weapon export

Diffie-Hellman Key Exchange

- start by sample two large primes: p, q s.t. q divides $p - 1$
- all math is done mod p (working in $\mathbb{Z}_{|}$)
- since q divides p, there exists a g s.t. $g^q = 1$, this will serve as the generator for a Group ($\mathbb{G} := g^a : a = 0, ..., q - 1$)

Diffie-Hellman Key Exchange



$\mathcal{A}$

$\mathcal{B}$

$\alpha \xleftarrow{R} \mathbb{Z}_q, u \leftarrow g^\alpha$

u

$\beta \xleftarrow{R} \mathbb{Z}_q, v \leftarrow g^\beta$

v

$w \leftarrow v^\alpha$

$w \leftarrow u^\beta$

$$w = v^\alpha = u^\beta = g^{\alpha\beta}$$

Diffie-Hellman Security

- Security rests on the difficulty of the discrete log problem
- over a cyclic group $\mathbb{G}$ it is mathematically hard to compute α given g^{α}, where g is a generator of $\mathcal{G}$
- this is further extended to: given (g^{α}, g^{β}) where g is a generator, $\alpha, \beta \xleftarrow{R} \mathbb{Z}_q$, it is hard to compute $g^{\alpha\beta}$