Random Escrow Project

Charles Smutz

Overview

- Introduction
  - Problem
  - Proposed Solution
- Implementation
- Proof of Concept
- Analysis

Problem

- Cryptography significantly hampers system monitoring/forensics

Solution

- OS level escrow of Random Number

Implementation

- OS level Tap into PRNG
- Decided to use Linux
- Interfaces into PRNG
  - get_random_bytes()
  - urandom_read()
  - random_read()
- Uses printk()
- Project Home Page
  - http://mason.gmu.edu/~csmutz/re/

Proof of Concept

- 1st step: Simplified PFS protocol (smotd)
  - 1a: Reconstruct session key
  - 1b: Decrypt encrypted message
- 2nd step: Real World
  - Real transport layer protocol (SSH)
  - Local Forensics (PGP)
- 3rd step: Address security of Escrow

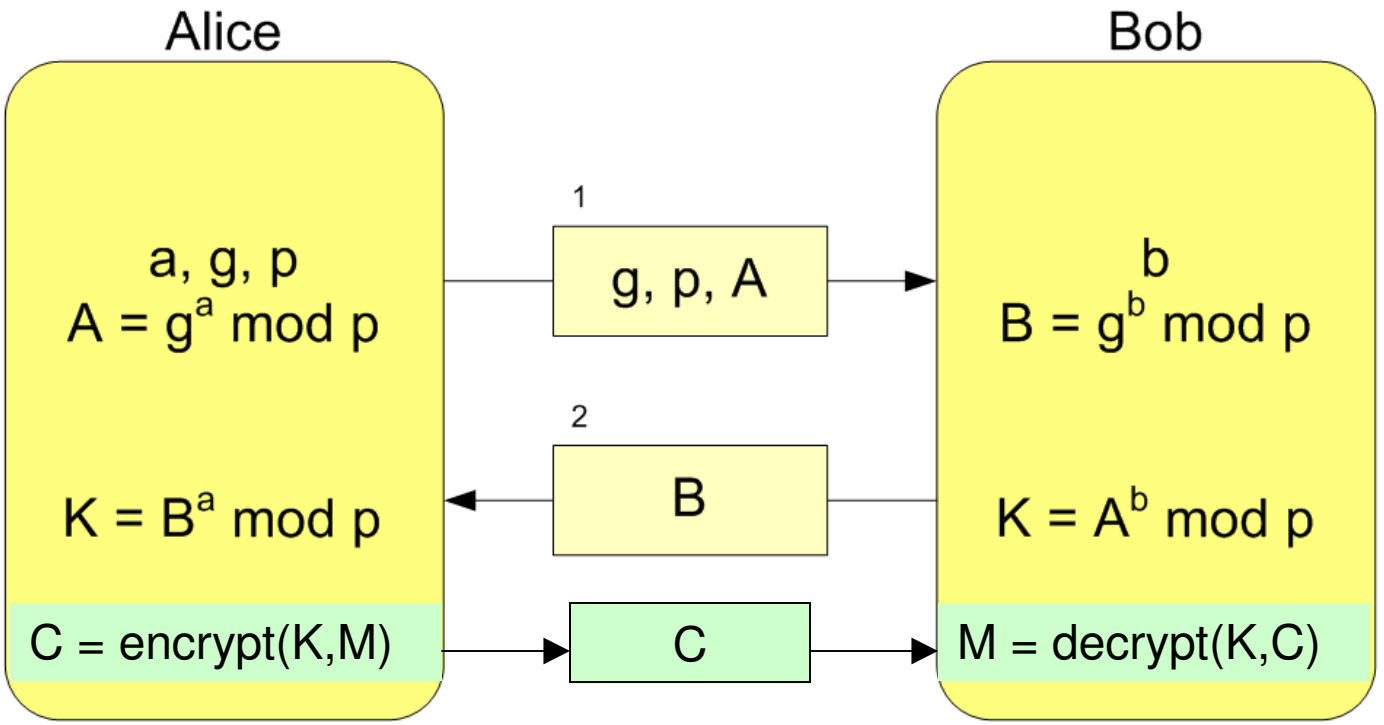
Proof of Concept: Step 1

- Given random number used to generate DH private key, and public key sent from other side, break session key
  - Challenge:
    - Random value -> Private Key
      - Duplication of Key Derivation Algorythm
  - Rest is capture and computation

Proof of Concept: Step1

- Secure Message of the Day Protocol
  - Client connects to server
  - Negotiate session key via DHE
  - Server sends MOTD (text) to client encrypted with session key
  - No authentication
  - PFS!

Proof of Concept: Step 1



$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

1. Intercept A via Packet Capture

2. Derive b from Random Escrow

3. Compute

Taken from: http://upload.wikimedia.org/wikipedia/commons/a/a3/Diffie-Hellman-Schl%C3%BCsselaustausch.png

Proof of Concept: Steps 2+3

- Challenge:
  - Duplicating Full protocols
    - Complexity

Analysis

- Issues:
  - Correlation of Escrowed Value to Use
  - Knowledge of Key Derivation

Questions?