Cryptographic Keys

Random numbers

Used in many places throughout cryptography
• A symmetric key is typically a random bit string
• For RSA, choose large numbers at random and then test for primality
• The Miller-Rabin primality test itself uses random values of \( a \)
• Lots of protocols involve random numbers

Required properties
• Randomness
• Unpredictability

Usually, true random numbers are not used
• Use pseudorandom number generators (PRNGs) instead
Example PRNGs:

a) Linear Congruential Generator

\[ X_{n+1} = (aX_n + c) \mod m \]

Need to choose parameters carefully

Desired properties:

1.

2.

3.

Suitable parameters:

Good for stats, simulation. What about cryptography?

- Not good at all…
b) Cryptographically generated random numbers

Idea: if an encryption algorithm is good, why not use it to generate a pseudorandom sequence?

E.g., Counter

E.g., DES (or any other good cipher) in OFB

E.g., ANSI X9.17 PRNG

E.g., PGP PRNG

E.g., Blum, Blum, Shub (BBS) Generator
Key Management (Chap. 7 [symm. env.]; Chap. 10 [asymm. env.])

A) A wants to talk to B in a symmetric environment. There are several alternatives for key distribution:

1. A physically delivers key
2. 3rd-party physically delivers key
3. Use previously-shared key to protect new key
4. Online Authority

B) A wants to talk to B in a public-key environment. There are several alternatives for key distribution:

1. Announcement
2. Directory
3. Online Authority
4. Certificates
5. D-H protocol