Public Key Cryptography

Hidden in plain sight

Confidentiality

- Three (randomized) algorithms
  - Generate: $G(1^k) \rightarrow K, K^{-1}$ (randomized)
  - Encrypt: $E(K,m) \rightarrow \{m\}_K$ (randomized)
  - Decrypt: $D(K^{-1}, \{m\}_K) \rightarrow m$
- Provides confidentiality: can't derive m from $\{m\}_K$ without K^{-1}
  - K can be made public: can't derive K^{-1} from K
  - Everyone can share the same K ("public" key)
- Encrypt must be randomized
  - Same plaintext sent multiple times must generate different ciphertexts
  - Otherwise can easily guess plaintext for small message space ("yes", "no")

Integrity: Signatures

- Three (randomized) algorithms
  - Generate: $G(1^k) \rightarrow K, K^{-1}$ (randomized)
  - Sign: $S(K^{-1}, m) \rightarrow \{m\}_{K^{-1}}$ (can be randomized)
  - Verify: $V(K, \{m\}_{K^{-1}}, m) \rightarrow$ {yes, no}
- Provides integrity like a MAC
  - Cannot produce valid $(\{m\}_{K^{-1}}, m)$ pair without K^{-1}
  - But only need K to verify, cannot derive K^{-1} from K
  - So K can be publicly known

Popular Public Key Algorithms

- Encryption: RSA, Rabin, ElGamal
- Signature: RSA, Rabin. ElGamal, Schnorr, DSA, ...
- Warning: message padding critically important
  - Basic idea behind RSA encryption is simple (modular exponentiation of large integers)
  - But simple transformations of message to numbers is not secure
- Many keys support both signing and encryption
  - But they use different algorithms
  - Common error: Sign by "encrypting" with private key

Example: RSA

- Generate private key K^{-1} and public key K
  - Choose two distinct prime numbers p, q
  - Compute $n = pq$
  - Compute (too complex for here) K and K^{-1} from p and q
- Advertise n and K as public key
  - $E(K,n,m) \rightarrow \{m\}_{K,}$ $\quad E(K,n,m) = m^K \, mod \, n$
  - $D(K^{-1},n, \{m\}_K) \rightarrow m_,$ $\quad D(K^{-1},n, \{m\}_K) = \{m\}_K{}^{K^{-1}} \, mod \, n$
- Can send n as cleartext: cannot derive p and q from n
  - If someone figured out how to quickly factor primes, it all crashes down
  - Inside NP, suspected to be outside P, but suspected to be not NP-complete

The Catch

- Cost of public key algorithms is significant

Algorithm	Encrypt	Decrypt	Sign	Verify
RSA 1024	0.08ms	1.46ms	1.48ms	0.07ms
RSA 2048	0.16ms	6.08ms	6.05ms	0.16ms
DSA 1024			0.45ms	0.83ms
LUC 2048	0.18ms	9.89ms	9.92ms	0.18ms
DLIES 2048	4.11ms	3.86ms		

- In contrast, symmetric algorithms can operate at line speed

http://www.cryptopp.com/benchmarks.html

Hybrid Schemes

- Use public key to encrypt symmetric key
- Negotiate secret session key
  - Use public key crypto to establish 4 symmetric keys
  - Client sends server $\{\{m_1\}_{K1}, MAC(K_2, \{m1\}_{K1})\}$
  - Server sends client $\{\{m_2\}_{K3}, MAC(K_4, \{m_2\}_{K3})\}$
- Common pitfall: signing underspecified messages
  - E.g., always specify intended recipient in signed messages
  - Should also specify expiration, or better yet fresh data
  - Otherwise like signing a blank check to anyone for all time...