Introduction to Public-Key Cryptography

Martin Stanek

Department of Computer Science
Comenius University
stanek@dcs.fmph.uniba.sk

Cryptology 1 (2020/21)

Content

Introduction 1

- problems with secret-key (symmetric) cryptography
  - encryption – all parties need to know the key
  - distributing the key
- idea of public-key cryptography
  - user generates a related pair of keys – public and private
  - private key can't be computed from the public key efficiently
- public-key / asymmetric schemes
  - encryption schemes
  - digital signature schemes
  - key agreement schemes/protocols
  - ...

Introduction 2

- public-key
  - used for encryption (in asymmetric encryption schemes), verification of signatures (in digital signature schemes), etc.
  - can be distributed freely, i.e. anyone can encrypt data for the user or verify user's signatures
  - How to ensure the authenticity of the public key? PKI?
- private-key
  - used for decryption (asymmetric encryption schemes), signing (digital signature schemes), etc.
  - should be kept private

Example with a phone book (encryption scheme)

- public key: large phone book (printed), sorted by names
- private key: phone book sorted by phone numbers (or a hash table)
- it should be "hard" to produce inversely sorted list from a printed book
- encryption (using the public key):
  - each letter is replaced by a randomly chosen phone number from the set corresponding to all names starting with the letter
  - HELLO
    $\longrightarrow$ **H**ender, Eric; **E**llison, Peter; **L**und, Aron; **L**orain, Beth; **O**mar, Lars
    $\longrightarrow$ 721-2876, 781-9817, 201-0928, 562-9873, 982-1982
- decryption:
  - binary search in the private-key phone book ... $O(\log n)$/letter
  - lookup in the private-key hash table ... $O(1)$/letter
  - without private-key: sequential search in public-key book ... $O(n)$/letter

Public-key encryption scheme (informally)

- 3-tuple $\langle \text{Gen}, \text{Enc}, \text{Dec} \rangle$ of algorithms:
  - Gen – probabilistic polynomial time algorithm; produces public and private key pair (pk, sk)
  - Enc – probabilistic polynomial time algorithm; computes ciphertext from plaintext and public key: $\text{Enc}_{\text{pk}}(m)$
  - Dec – (usually deterministic) polynomial time algorithm; computes plaintext from ciphertext and private key: $\text{Dec}_{\text{sk}}(c)$
- Requirements:
  - correctness: $\forall (\text{pk}, \text{sk}) \leftarrow \text{Gen}() \; \forall m : \; \text{Dec}_{\text{sk}}(\text{Enc}_{\text{pk}}(m)) = m$
  - efficiency: (probabilistic) polynomial time
  - security

Security of the PK encryption scheme (discussion)

How to define security?

1. inability to extract private-key from public-key
   - it says *nothing* about confidentiality of plaintext
   - consider a scheme where plaintext is a part of the ciphertext
2. inability to decrypt ciphertext without private-key
   - each ciphertext / what fraction of all ciphertexts?
   - what part of ciphertext?
   - it says *nothing* about recognizing what is not encrypted, about meaningfully modifying ciphertext, etc.
3. inability to distinguish encryptions of different plaintexts

more about security later …

Hard problems in cryptography

- public-key cryptography is based on hard problems
- the most common/popular are:
  - factorization – RSA problem, Square root problem, Quadratic residuosity problem
  - discrete logarithm (in various groups) – DLOG, Decisional/Computational Diffie-Hellman problem
- other problems:
  - lattices: SVP (Shortest vector problem), LWE (Learning with error), ...
  - pairings: Bilinear Diffie-Hellman
  - coding: Syndrome Decoding
  - Multivariate Equation Solving, ...

Factorization

- usually: $n = p \cdot q$ (product of two large primes); compute p and q
- How hard is this problem (generally)?
- best general algorithm: General Number Field Sieve (GNFS)
- heuristic complexity: $\exp\left((\sqrt[3]{64/9} + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}\right)$
- equivalent/comparable key sizes
  (NIST SP 800-57 part 1 rev. 5, 2020):

symmetric	integer factorization
80	1024
112	2048
128	3072
192	7680
256	15360

- various methods are compared at www.keylength.com

Discrete logarithm

- let g be a generator of group G; given $y \in G$ compute x: $g^x = y$
- easy/hard depending on group
- frequently used groups:
  - multiplicative group (or some subgroup) modulo prime: $(\mathbb{Z}_p \setminus \{0\}, \cdot)$
  - points on an elliptic curve (with addition)
- equivalent/comparable key sizes (NIST SP 800-57 part 1 rev. 5):

symmetric	finite field group	elliptic curves
80	1024 (160)	160-223
112	2048 (224)	224-255
128	3072 (256)	256-383
192	7680 (384)	384-511
256	15360 (512)	512+

Post-quantum cryptography

- assumption: large-scale quantum computers
- Shor's algorithm – polynomial time algorithm for integer factorization
  - a variant for discrete logarithm problem (for integer arithmetic, elliptic curves)
- alternatives: code-based, lattice-based, hash-based, multivariate cryptosystems

Post-quantum cryptography – future

- NSA announced preliminary plans for transitioning to quantum resistant algorithms (August 2015):
  IAD will initiate a transition to quantum resistant algorithms in the not too distant future. Based on experience in deploying Suite B, we have determined to start planning and communicating early about the upcoming transition to quantum resistant algorithms. Our ultimate goal is to provide cost effective security against a potential quantum computer.
- NIST (August 2016):
  The National Institute of Standards and Technology (NIST) is requesting comments on a new process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms.
  - https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
  - 69 submissions (Nov 2017), 3-round analysis
  - 29 candidates in round 2 (Jan 2019)
  - 7 candidates in round 3 and 8 “alternate” candidates (July 2020)
  - draft standards expected in 2022/2024