CS 120/CSCI E-177: Introduction to Cryptography

Salil Vadhan and Alon Rosen Nov. 14, 2006

Lecture Notes 14:

Public-Key Encryption

Recommended Reading.

- Katz–Lindell, Sections 9.1–9.3, 9.5.2, 9.5.4, 8.4.5

1 Setting

- Can parties communicate privately *without meeting in advance*? One of the drawbacks of private-key encryption is the exchange of the secret key: the parties have to meet in advance or use some more secure channel.
- Classical view: ability to encrypt ≡ ability to decrypt ≡ possession of key
- Diffie–Hellman '76: *public-key encryption* — separate encryption & decryption keys
  - *public key* = encryption key, anyone can encrypt since the public key is published in some public directory
  - *secret key* = decryption key
  - secret key $\mapsto$ public key should be infeasible
  - We can imagine that there exists a public directory containing everyone's public key: pk_{Alice},pk_{Bob},... To send a message m to Alice, we get pk_{Alice} from the directory and send $E_{pk_{\text{Alice}}}(m)$. The real implementation of the public-key infrastructure requires care. For instance, how can we be sure that we access the *real* public directory?
- **Definition 1** *A* public-key encryption scheme *consists of three polynomial-time algorithms* (G, E, D)*, as follows:*
  - *The* key generation *algorithm* G *is a randomized algorithm that takes a security parameter* 1^n *as input returns a pair* (pk, sk)*, where* pk *is the* public key *and* sk *is the* secret key*; we write* $(pk, sk) \stackrel{R}{\leftarrow} G(1^n)$.
  - *The* encryption *algorithm* E *is a* stateless *randomized algorithm that takes the public key* pk *and a* plaintext *(aka* message*)* m *and outputs a ciphertext* c*; we write* $c \stackrel{R}{\leftarrow} E_{pk}(m)$.
  - *The* decryption *algorithm* D *is a deterministic algorithm that takes the secret key* sk *and a ciphertext* c *and returns a plaintext* $m = D_{sk}(c)$.

  Associated with the scheme is a *plaintext space* $\mathcal{P}_n$ from which m is allowed to be drawn. The message space is a set, often the set of strings of a given length. We require $D_{sk}(E_{pk}(m)) = m$ for all $m \in \mathcal{P}$ and all $(pk, sk) \leftarrow G(1^n)$.

2 Security

- All definitions of security in the private-key case extend naturally to public-key case — just provide the adversary with the public key.

  Definition 2 (indistinguishability for public-key encryption) *Let (G, E, D) be a public-key encryption scheme over $\mathcal{P} = \bigcup_n \mathcal{P}_n$. (G, E, D) has* indistinguishable encryptions *if for every (nonuniform) PPT A, there is a negligible function ε such that for all $m_0, m_1 \in \mathcal{P}_n$ such that $\|m_0\| = \|m_1\|$,*

  $$|\Pr[A(1^n, pk, E_{pk}(m_0)) = 1] - \Pr[A(1^n, pk, E_{pk}(m_1)) = 1]| \leq \varepsilon(n),$$

  where the probabilities above are taken over $(pk, sk) \stackrel{R}{\leftarrow} G(1^n)$, the coin tosses of E_{pk}, and the coin tosses of A.

- All of the relations we have established between notions of security extend to the public-key case, such as the equivalence of indistinguishability and semantic security. Hence public-key encryptions satisfying the above definition are often referred to as *semantically secure* encryption schemes in the literature (and in KL).

- **Lemma 3** *If public-key encryption scheme (G, E, D) has indistinguishable encryptions in the sense of Definition 1, then it satisfies indistinguishability under chosen plaintext attack (and hence satisfies multiple-message indistinguishability).*

  Proof Sketch: Encryption oracle cannot help adversary, because the adversary can encrypt on its own. □

- Corollaries:
  - No deterministic public-key encryption scheme can be secure.
  - No public-key encryption scheme can be perfectly (or even statistically) secure.

3 Trapdoor Permutations

Definition 4 *$\mathcal{F} = \{f_k : D_k \to D_k\}_{k \in \mathcal{K}}$ is a* collection of trapdoor permutations *if each f_k is a permutation and:*

1. *There is a PPT $G(1^n)$ that outputs a pair (k, t), where $k \in \mathcal{K}$ is the (public)* key *(or* index*) and t is the* trapdoor*.*
2. *Given k, one can sample uniformly from D_k in polynomial time.*
3. *Given k and $x \in D_k$, one can evaluate $f_k(x)$ in polynomial time.*
4. *For every (nonuniform) PPT A, there is a negligible function ε such that*

   $$\Pr\left[A(1^n, K, f_K(X)) \in f_K^{-1}(f_K(X))\right] \leq \varepsilon(n) \qquad \forall n$$

   where the probability is taken over $(K, T) \stackrel{R}{\leftarrow} G(1^n)$, $X \stackrel{R}{\leftarrow} D_K$, and the coin tosses of A.

5. *Given t and $y \in D_k$, one can evaluate $f_K^{-1}(y)$ in polynomial time.*

Examples: there are many fewer candidates than OWF

1. RSA: the trapdoor for $f_{N,e}$ is $t = d$ such that $ed \equiv 1 \pmod{\phi(N)}$.
   Inverse map:
   Key generation:
2. Rabin: we have to restrict to the case where Rabin's functions f_N are permutations, i.e. $N = pq$ for $p, q \equiv 3 \bmod 4$ and the domain is QR_N.
   Trapdoor:
   Inverse map:

4 Encryption from Trapdoor Permutations

Direct use of trapdoor permutation family $\mathcal{F}$

- Randomly select f_k with trapdoor t.
- $pk = k$, $sk = t$.
- $E_{pk}(x) = f_k(x)$, $D_{sk}(y) = f_k^{-1}(y)$.
- e.g. Plain RSA: $E_{N,e}(x) = x^e \bmod N$, $D_{N,d}(y) = y^d \bmod N$.
- Insecure.

Better: use hardcore bits

An encryption scheme for message space $\{0,1\}$.

- Randomly select f_k with trapdoor t.
- $pk = k$, $sk = t$.
- $E_{pk}(m)$: Choose $x \xleftarrow{\mathrm{R}} D_k$. Output $(f_k(x), b_k(x) \oplus m)$.
- $D_{sk}(y, c) = (b_k(f_k^{-1}(y)) \oplus c)$.

Theorem 5 *Above public-key encryption scheme is secure if $\{f_k\}$ is a collection of trapdoor permutations with hardcore bits $\{b_k\}$.*

Proof Sketch:

$$\begin{aligned}
(PK, E_{PK}(0)) &\equiv (K, f_K(X), b_K(X)) \\
&\overset{\mathrm{c}}{\equiv} (K, f_K(X), R) \\
&\equiv (K, f_K(X), \overline{R}) \\
&\overset{\mathrm{c}}{\equiv} (K, f_K(X), \overline{b_K(X)}) \\
&\equiv E_{PK}(1)
\end{aligned}$$

□

A more efficient construction

- Randomly select f_k with trapdoor t.
- $pk = k$, $sk = t$.
- $E_{pk}(m)$ for $m \in \{0,1\}^\ell$:
  1. Choose $x \xleftarrow{R} D_k$.
  2. Compute $G_{k,\ell}(x) = b_k(x)b_k(f_k(x))b_k(f_k(f_k(x)))\cdots b_k(f_k^{\ell-1}(x))$.
  3. Output $\big(f_k^\ell(x), G_{k,\ell}(x) \oplus m\big)$.

This construction is not more efficient in terms of computation because we still have to evaluate f_k for each bit of the message but it is more efficient in terms of communication since the ciphertext is shorter.

Theorem 6 *Above public-key encryption scheme is secure if $\{f_k\}$ is a collection of trapdoor permutations with hardcore bits $\{b_k\}$.*

Proof Sketch: $(K, f_K^\ell(X), G_{K,\ell}(X)) \stackrel{c}{\equiv} (K, f_K^\ell(X), G_{K,\ell}(X))$. cf. construction of a PRG that stretches by an arbitrary polynomial. □

Blum-Goldwasser Encryption

This encryption scheme uses Rabin's functions: p, and q are n-bit primes, such that $p \equiv q \equiv 3 \bmod 4$ and $N = pq$. The public key is $pk = N$ and the secret key is $sk = \{p, q\}$.

Encryption $E_{pk}(m)$:

- Choose $x \xleftarrow{R} \mathrm{QR}_N$.
- Let $r = \mathrm{lsb}(x)\mathrm{lsb}(x^2)\cdots\mathrm{lsb}(x^{2^{\ell-1}} \bmod N)$ where $\ell = \|m|$.
- Output $(x^{2^\ell} \bmod N, r \oplus m)$.

Decryption: given the factorization of N, how do we compute x from $x^{2^\ell} \bmod N$ efficiently?

5 Schemes Based on Discrete Log

- *Key Exchange*: get rid of public directory, agree on key "on the fly".
- *Diffie–Hellman Key Exchange:*
  - Prime p, generator g fixed.
  - Alice chooses $x \xleftarrow{R} \mathbb{Z}_{p-1}$, sends g^x to Bob.
  - Bob chooses $y \xleftarrow{R} \mathbb{Z}_{p-1}$, sends g^y to Alice.
  - Both can compute the *Diffie–Hellman Key* $k = g^{xy}$.
- Security of DH Key Exchange:

  - Can be broken if discrete log problem easy, but seems to require *Computational Diffie–Hellman assumption*: given p, g, g^x, g^y, hard to compute g^{xy} (with nonnegligible probability).
  - If Alice & Bob want to use g^{xy} as a key, actually need *Decisional Diffie–Hellman assumption*: (P, G, G^X, G^Y, G^{XY}) computationally indistinguishable from (P, G, G^X, G^Y, G^Z), where Z is chosen randomly independently form X and Y. (not quite true as stated, should take g to be a random generator of large subgroup of $\mathbb{Z}_p^*$ of prime size, e.g. of QR_p if $p = 2q + 1$ for prime q. See KL §8.4.5.)
  - DH Key Exchange insecure against active adversary: person-in-the-middle attack.
  - Properly defining security of key exchange vs. active adversaries very subtle!
- Variants of DH Key Exchange used in practice, e.g. cell phones.
- *El Gamal Public-Key Encryption Scheme:*
  - Replace interaction in DH Key Exchange with public directory.
  - Secret Key: (p, g, x), Public Key: $(p, g, \hat{x})$, where $\hat{x} = g^x \bmod p$.
  - $E(m)$ (for $m \in \mathbb{Z}_p^*$):
    1. Choose $y \xleftarrow{\mathrm{R}} \mathbb{Z}_{p-1}$
    2. Compute $\hat{y} = g^y \bmod p$.
    3. Compute $w = \hat{x}^y \cdot m \bmod p$. (like using $\hat{x}^y = g^{xy}$ as a one-time pad)
    4. Output $(\hat{y}, w)$.
  - Secure under Decisional Diffie–Hellman Assumption (with same caveats as above; see KL)
- Cramer-Shoup encryption scheme: strengthened form of El Gamal with security against *chosen-ciphertext attack* (and efficiency within a small constant factor)
- DH Key Exchange, El Gamal make sense using any cyclic group in which discrete logs are hard (e.g. also elliptic curves).
- There are other public-key encryption schemes based on: lattices, knapsack problem (related to subset sum), error-correcting codes. . . (some proven based on complexity assumptions, some not).