Crypto basics

Subhashis Banerjee, Subodh Sharma

Department of Computer Science and Engineering
IIT Delhi

November 3, 2020

Private-key encryption

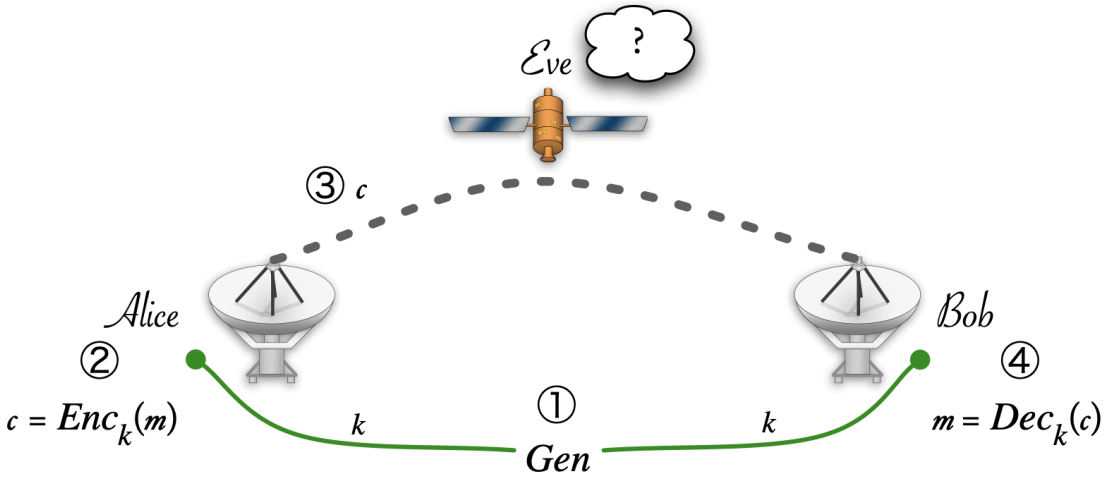

Figure 2.1: Illustration of the steps involved in private-key encryption. First, a key k must be generated by the Gen algorithm and privately given to Alice and Bob. In the picture, this is illustrated with a green "land-line." Later, Alice encodes the message m into a ciphertext c and sends it over the insecure channel—in this case, over the airwaves. Bob receives the encoded message and decodes it using the key k to recover the original message m. The eavesdropper Eve does not learn anything about m except perhaps its length.

(Source: *A Course in Cryptography* by Rafael Pass and Abhi Shelat)

Private-key encryption

Security by obscurity: All of (Gen, Enc, Dec) and k are *private*.

Modern cryptography: (Gen, Enc, Dec) are *public*. Only k is *private*.
Follows *Kerchoff's (1884) principle*.

Kerchoff's (1884) principle implies that at least GEN must be randomized.

Private-key encryption

The triplet of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is called a *private-key encryption scheme* over message space $\mathcal{M}$ and key space $\mathcal{K}$ if:

1. Gen (called the key generation algorithm) is a randomized algorithm that returns a key k such that $k \in \mathcal{K}$. We denote by $k \leftarrow \mathsf{Gen}$ the process of generating a key k.
2. Enc (called the encryption algorithm) is a potentially randomized algorithm that on input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, outputs a ciphertext c. We denote by $c \leftarrow \mathsf{Enc}_k(m)$ the output of Enc on input key k and message m.
3. Dec (called the decryption algorithm) is a deterministic algorithm that on input a key k and a ciphertext c outputs a message $m \in \mathcal{M} \cup \perp$.
4. $\forall m \in \mathcal{M}$, $\Pr[k \leftarrow \mathsf{Gen} : \mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] = 1$.

Does not specify any privacy *or* security *properties.*

The Caesar cipher

$$
\begin{aligned}
\mathcal{M} &= \{A, B, \ldots, Z\}^* \\
\mathcal{K} &= \{0, 1, \ldots, 25\} \\
\text{Gen} &= k \text{ where } k \xleftarrow{r} \mathcal{K} \\
\text{Enc}_k(m_1 \ldots m_n) &= c_1 \ldots c_n \text{ where } c_i = m_i + k \bmod 26 \\
\text{Dec}_k(c_1 \ldots c_n) &= m_1 \ldots m_n \text{ where } m_i = c_i - k \bmod 26
\end{aligned}
$$

- The Caesar cipher is a *private-key encryption* scheme.
- *Attack*: Just try all 26 shifts.

The substitution cipher

$$
\begin{aligned}
\mathcal{M} &= \{A, B, \ldots, Z\}^* \\
\mathcal{K} &= \text{ the set of permutations of } \{A, B, \ldots, Z\} \\
\text{Gen} &= k \text{ where } k \xleftarrow{r} \mathcal{K} \\
\text{Enc}_k(m_1 \ldots m_n) &= c_1 \ldots c_n \text{ where } c_i = k(m_i) \\
\text{Dec}_k(c_1 \ldots c_n) &= m_1 \ldots m_n \text{ where } m_i = k^{-1}(c_i)
\end{aligned}
$$

- The substitution cipher is a *private-key encryption* scheme.
- There are now 26! possible keys.
- *Attack*: Do a frequency analysis of alphabets in english text.

‘Crypto cycle’

1. **A**, the “artist”, invents a security/privacy scheme.
2. **A** claims (or even mathematically proves) that known attacks do not work.
3. The security/privacy scheme gets employed widely (often in critical situations).
4. The scheme eventually gets broken by improved attacks.
5. Restart, usually with a patch to prevent the previous attack.

From Julius Caesar to the government of India, also IT industry.

Modern cryptography and provable security

- Provide mathematical definitions of security.
- Provide precise mathematical assumptions (e.g. "factoring is hard", where *hard* is formally defined). These can be viewed as axioms.
- Provide proofs of security, i.e., prove that, if some particular scheme can be broken, then it contradicts an assumption (or axiom). In other words, if the assumptions were true, the scheme cannot be broken.

Other examples

Secure 2 party computation Allows two parties A and B with private inputs a and b respectively, to compute a function $f(a, b)$ that operates on joint inputs a, b while guaranteeing the same correctness and privacy as if a trusted party had performed the computation for them, even if either A or B try to deviate from the proscribed computation in malicious ways.

Zero-knowledge proofs Allows Alice to convince Bob that some statement is true; without compromising privacy. For instance, Alice may want to convince Bob that a number N is a product of two primes p, q without revealing p or q.

Possible notions of privacy or security

- The adversary cannot learn (all or part of) the key from the ciphertext. (**Too weak**)
- The adversary cannot learn (all, part of, any letter of, any function of, or any partial information about) the plaintext. (**Too strong**)
- Given some a priori information, the adversary cannot learn any additional information about the plaintext by observing the ciphertext. (**Differential**)

Shannon secrecy (1949)

$(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a private-key encryption scheme that is *Shannon-secret with respect to the distribution* $\mathcal{D}$ over the message space $\mathcal{M}$ if, for all $m' \in \mathcal{M}$ and for all c,

$$\begin{aligned} & \Pr\left[k \leftarrow \mathsf{Gen}; m \leftarrow \mathcal{D} : m = m' \mid \mathsf{Enc}_k(m) = c\right] \\ = \quad & \Pr\left[m \leftarrow \mathcal{D} : m = m'\right] \end{aligned}$$

An encryption scheme is said to be *Shannon secret* if it is Shannon secret with respect to all distributions $\mathcal{D}$ over $\mathcal{M}$.

Perfect secrecy

$(\mathcal{M}, \mathcal{K}, \text{Gen}, \text{Enc}, \text{Dec})$ is said to be a private-key encryption scheme that is *perfectly secret* if, for all $m_1, m_2 \in \mathcal{M}$, and for all c,

$$\Pr[k \leftarrow \text{Gen} : \text{Enc}_k(m_1) = c] = \Pr[k \leftarrow \text{Gen} : \text{Enc}_k(m_2) = c]$$

- *Shannon secrecy* $\Leftrightarrow$ *perfect secrecy*

One-time pad

$$
\begin{aligned}
\mathcal{M} &= \{0,1\}^n \\
\mathcal{K} &= \{0,1\}^n \\
\text{Gen} &= k = k_1 k_2 \ldots k_n \leftarrow \{0,1\}^n \\
\text{Enc}_k(m_1 m_2 \ldots m_n) &= c_1 c_2 \ldots c_n \text{ where } c_i = m_i \oplus k_i \\
\text{Dec}_k(c_1 \ldots c_n) &= m_1 \ldots m_n \text{ where } m_i = c_i \oplus k_i
\end{aligned}
$$

- The One-Time Pad is a *perfectly secure* private-key encryption scheme.

  For any $c, m \in \{0,1\}^n$, $\Pr[k \leftarrow \{0,1\}^n : \text{Enc}_k(m) = c] = 2^{-n}$
- So, *perfect secrecy* is obtainable.
- But at what cost?

Shannon's theorem and fallouts

- If $(\mathcal{M}, \mathcal{K}, \text{Gen}, \text{Enc}, \text{Dec})$ is a perfectly secret private-key encryption scheme, then $|\mathcal{K}| \geq |\mathcal{M}|$.
- If $|\mathcal{M}| > |\mathcal{K}|$, then there exist $m_1, m_2 \in \mathcal{M}$ and $\epsilon > 0$ such that

$$\Pr[k \leftarrow \mathcal{K}; \text{Enc}_k(m_1) = c : m_1 \in \mathbf{Dec}(c)] = 1$$
$$\Pr[k \leftarrow \mathcal{K}; \text{Enc}_k(m_1) = c : m_2 \in \mathbf{Dec}(c)] = 1 - \epsilon$$

where

$$\mathbf{Dec}(c) = \{m \mid \exists k \in \mathcal{K} \text{ such that } m = \text{Dec}_k(c)\}; |\mathbf{Dec}(c)| \leq |\mathcal{K}|$$

- Then there is an attack. Suppose Alice picks m from $\{m_1, m_2\}$ uniformly and sends to Bob. Eve checks if $m_2 \in \mathbf{Dec}(c)$. If yes, she makes a random guess, otherwise she guesses $m = m_1$. She succeeds with probability

$$\Pr[m = m_2](1/2) + \Pr[m = m_1](\epsilon \cdot 1 + (1 - \epsilon) \cdot (1/2)) = 1/2 + \epsilon/4$$

Stronger version of Shannon's theorem

- If $(\mathcal{M}, \mathcal{K}, \text{Gen}, \text{Enc}, \text{Dec})$ is a perfectly secret private-key encryption scheme where $\mathcal{M} = \{0,1\}^n$ and $\mathcal{K} = \{0,1\}^{n-1}$, then

$$\Pr\left[k \leftarrow \mathcal{K}; \text{Enc}_k(m_1) = c : m_2 \in \textbf{Dec}(c)\right] \leq 1/2$$

- With $\epsilon = 1/2$, the probability of Eve's success is $5/8$.
- However, it is computationally hard. If $\mathcal{K} = \{0,1\}^n$, then the attack requires $O(2^n)$ decryptions.

Efficient computation and polynomial-time bounded adversary

- A triplet of algorithms Gen, Enc, Dec) is an *efficient private-key encryption scheme* if all three are randomized polynomial time algorithms (probabilistic polynomial time Turing machines, or p.p.t.).
- We allow the adversary to use random coins and require that the adversary's running time is bounded by a polynomial. Program size can also increase polynomially with input length (non-uniform).

One-way functions

- Basic desiderata
  - it must be feasible to generate c given m and k, but
  - it must be hard to recover m and k given only c
- A function f is *worst-case one-way* if
  1. there is a p.p.t $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$
  2. there is no adversary $\mathcal{A}$ such that $\forall x \Pr\left[\mathcal{A}(f(x)) \in f^{-1}(f(x))\right] = 1$
- If **NP** $\not\subset$ **BPP** then worst-case one-way functions must exist.
- However, there may be pathological one-way functions that are easy to invert for *most* x, but every machine fails to invert $f(x)$ for infinitely many x's.

One-way functions

- A function $\epsilon(n)$ is a **negligible function** if for every c, there exists some n_0 such that $n > n_0$, $\epsilon(n) < 1/n^c$.
- A function is *strong one-way* if any efficient attempt to invert f on random input succeeds with only negligible probability. That is, for any adversary $\mathcal{A}$ there exists a negligible function ϵ such that for any input of length n

$$\Pr[x \leftarrow \{0,1\}^n; y \leftarrow f(x) : f(\mathcal{A}(1^n, y)) = y] \leq \epsilon(n)$$

- A function is *weak one-way* if $\exists$ a polynomial function $q(n) : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$\Pr[x \leftarrow \{0,1\}^n; y \leftarrow f(x) : f(\mathcal{A}(1^n, y)) = y] \leq 1 - \frac{1}{q(n)}$$

(all efficient attempts to invert will fail with some non-negligible probability)

Multiplication, primes, factoring

- Is $f_{\text{mult}} : \mathbb{N}^2 \to \mathbb{N}$ one-way?

$$f_{\text{mult}}(x, y) \begin{cases} 1 & \text{if } x = 1 \vee y = 1 \\ x \cdot y & \text{otherwise} \end{cases}$$

- Let

$$\Pi_n = \{q \mid q < 2^n \text{ and } q \text{ is prime}\}$$

- **Factoring assumption:** For every adversary $\mathcal{A}$, there exists a negligible function ϵ such that

$$\Pr\left[p \leftarrow \Pi_n; q \leftarrow \Pi_n; N \leftarrow pq : \mathcal{A}(N) \in \{p, q\}\right] < \epsilon(n)$$

- The best provable algorithm runs in time $2^{O((n \log n)^{1/2})}$ and the best heuristic algorithm runs in time $2^{O(n^{1/3} \log^{2/3} n)}$.
- *If the factoring assumption is true, then f_{mult} is a weak one-way function.*

The *prime* story

- There are *infinitely many* primes (Euclid, 300 BC).
- "Primes are *dense*". The number of primes $\leq N$, $\pi(N) \simeq N/\ln N$, as $N \to \infty$ (**Prime number theorem** - Gauss, Chebyshev).
- There are efficient algorithms to test primality (Miller-Rabin, Solovay-Strassen, Agrawal-Kayal-Saxena (?)).
- Factorization is *not easy*. Kleinjung and 12 colleagues had this to say in 2010 about the RSA-768 challenge modulus.

  We spent half a year on 80 processors on polynomial selection. This was about 3% of the main task, the sieving, which was done on many hundreds of machines and took almost two years. On a single core 2.2 GHz AMD Opteron processor with 2 GB RAM per core, sieving would have taken about fifteen hundred years.

Factoring and order finding

- (**Theorem**) Suppose N is a composite number, and x is a non-trivial solution to the equation $x^2 = 1 \pmod{N}$ in the range $1 \leq x \leq N$, that is, neither $x = 1 \pmod{N}$ nor $x = N - 1 = -1 \pmod{N}$. Then at least one of $\gcd(x-1, N)$ and $\gcd(x+1, N)$ is a non-trivial factor of N.
- (**Theorem**) Suppose $N = p_1^{\alpha_1} \ldots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq N-1$ and x is co-prime to N. Let r be the order of x modulo N. Then

$$\Pr\left[r \text{ is even and } x^{r/2} \neq \pm 1 \pmod{N}\right] \geq 1 - \frac{1}{2^m}$$

Factoring and discrete log

Summary

- A private-key encryption system has *perfect secrecy* if given ciphertext c, all messages are equally probable.
- *Perfect secrecy* $\Leftrightarrow$ *Shannon secrecy*.
- Perfect secrecy is attainable (one-time pad), but it requires $|\mathcal{K}| \geq |\mathcal{M}|$. Impractical in most situations.
- Often suffices to seek security against a *polynomial-time bounded adversary*. [against a non-uniform probabilistic polynomial-time Turing machine (n.u. p.p.t.)]
- Such security usually derived from one-way functions; from the assumed hardness of *factoring* or *order finding* or *discrete log* or some such.