# Security (Cryptographic) Protocols

**Definition.** A prescribed sequence of interactions between entities designed to achieve goals that may include:

- *Authentication* of entities;
- Establishing *session keys* between entities;
- Ensuring *secrecy*, *anonymity*, *nonrepudiaiton*, etc.

The protocols involve exchange of messages between entities, and use various cryptographic mechanisms (e.g., encryption, hash functions, digital signatures, and timestamps.)

*Security Protocols are Notoriously Difficult to Analyze!!*

- Properties are *extremely* subtle;
- Protocols run in a *complex and hostile* environment;
- Hard to capture the *power of the adversary.*
- *Concurrent protocols* are generally hard to analyze.

# Example:Needham Schroeder Secret Key protocol

This is one of the earliest protocols. Has a subtle vulnerability (to be discussed later). Forms the basis of the Kerberos[1] authentication protocol. Uses purely symmetric encryption algorithms. Designed to enable Alice and Bob to set up a secure channel of communication with the help of a trusted server, Trent.

Both Alice and Bob share a private, long-term key with Trent, but cannot communicate with each other directly and securely.

Alice wants to initiate a private communication with Bob, and wishes to avoid sending all message through Trent (thus, Trent should not become

[1] An authentication service developed as part of Project Atehna at MIT. Assumes an open distributed environment. Servers have to restrict access to authorized users and authenticate requests for services.

a bottleneck.) The idea is to use TRENT's help to generate a *fresh* session key between for ALICE and BOB.

Why not assign secret key for communication between ALICE and BOB? This will require roughly $N^2$ keys for communication between each two of $N$ entities, while most of the keys may never be used. In addition, the set of users may not be known in advance; in fact, it may not be necessarily static.

## The protocol

| | |
|---|---|
| **Message 1.** | $A \longrightarrow T : A.B.N_a$ |
| **Message 2.** | $T \longrightarrow A : \{N_a.B.k_{ab}.\{k_{ab}.A\}_{K_B}\}_{K_A}$ |
| **Message 3.** | $A \longrightarrow B : \{k_{ab}.A\}_{K_B}$ |
| **Message 4.** | $B \longrightarrow A : \{N_b\}_{k_{ab}}$ |
| **Message 5.** | $A \longrightarrow B : \{N_b - 1\}_{k_{ab}}$ |

**Message 1:** ALICE tells TRENT she wants to communicate with BOB and supplies a fresh nonce;

**Note:** Secrecy and authentication are not breached, but denial of service attack is possible. ALICE's message is sent in the clear. It is assumed that it's okay for intruder to learn that ALICE wishes to communicate with BOB. Intruder can alter message (no integrity) but ALICE will notice it in next step.

**Message 2:** TRENT sends to ALICE an encrypted message that includes her request, the new key, and a message to BOB (encrypted in BOB's secret key), that includes the new key.

**Note:** This message is a encrypted with ALICE's key, thus it is guaranteed to have originated with TRENT. ALICE should check that it is indeed her request that TRENT responded to. ALICE cannot decipher (read) that portion of the message that is encrypted with BOB's key, however, she trusts TRENT that it is indeed of the correct format (contains ALICE's name and the new session key.

**Message 3:** ALICE sends BOB the portion of TRENT's message that is intended for BOB.

**Note:** When BOB receives this message he can decrypt it and reveal the new session key and the identity of the party to the new session.

**Message 4:** BOB sends ALICE a fresh nonce encrypted with the new key supplied by TRENT.

**Note:** Upon receipt, ALICE knows that BOB knows new session key. But BOB does know know ALICE knows that he knows ...

**Message 5:** ALICE decrements BOB's nonce, encrypts it with the new key, and sends it to BOB.

**Note:** The decrement is to avoid replay of message 4. When BOB receives this message, he knows that ALICE knows that he knows the new key, and a new session can resume. This only works if block ciphers are used, since for stream ciphers it may be easy to generate $\{m-1\}_k$ given $\{m\}_k$.

# Needham-Schroeder Public Key Protocol

This is a protocol that dates back to 1978. It is described in Figure 1.

An intended run of the Needham-Schroeder is in Figure 2.

The protocol, however, has an undesirable run described in Figure 3 that was discovered by Lowe in 1995.

Who was duped? Not ALICE, since she meant to share $N_1$, $N_2$ with $P$. BOB however, thinks he shares $N_1$, $N_2$ only with ALICE. Thus secrecy failed ($P$ knows values), authentication failed (ALICE had no run with BOB.)

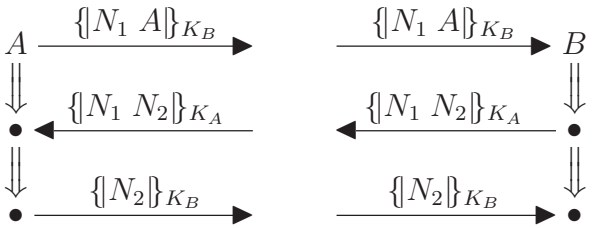


Figure 1: Needham-Schroeder Public Key Protocol

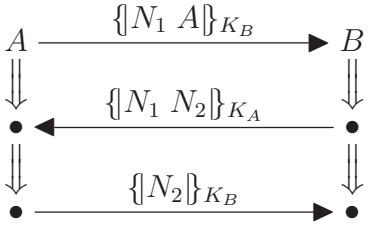


Figure 2: An Intended Run of the Protocol

How? ALICE offered $P$ a service. She gave $P$ nonce $N_1$, and promised to translate $\{\!|N_1, N|\!\}_{K_A}$ to $\{\!|N|\!\}_{K_P}$
An "unintended service": Attacker needs to compute some value ($N_2$ in this case), but legitimate party creates such a value.

# Functions

A function $f$ is a mapping between a set $X$ and a set $Y$, such that every element in $X$ has *precisely* one $Y$-element mapped to it. $X$ is the *domain* of the function, and $Y$ is the *co-domain*.

If $x$ is an element of $X$ (written as $x \in X$) then its *image* under $f$, denoted by $f(x)$, is the $Y$ element that $f$ maps to it.

We usually write $f\colon X \to Y$ to denote that $f$ is a function with domain $X$ and a co-domain $Y$. For $y \in Y$, the *pre-image* of $y$ is *some* $x \in X$ such that $f(x) = y$. Note that the pre-image of an element is not always defined.

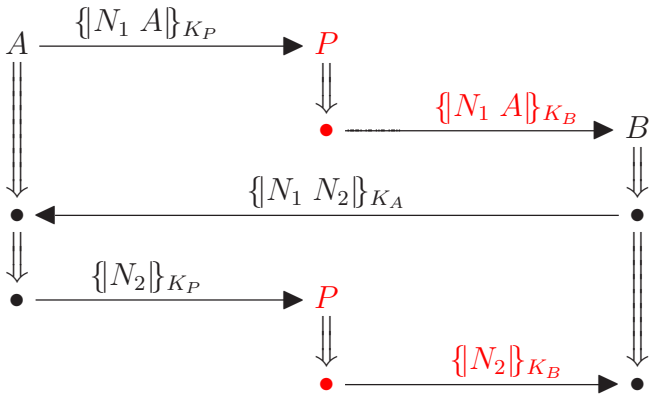


Figure 3: An Undesirable Run

A many-to-one function A 1–1 function

The set of all $Y$ elements that have at least one pre-image are called the *image of* $f$.

A function is 1–1 (*one-to-one*) if each element in the co-domain is the image of *at most* one element in the domain.

A function is *onto* if each element in the co-domain is the image of *at least* one element in the domain.

A function that is both 1–1 and onto is called a *bijection*.

## Examples of Functions

Note that if $f\colon X \to Y$ is 1–1, then $f\colon X \to \mathrm{Im}(Y)$ is a bijection. Also, if both $X$ and $Y$ are have the same number of elements, then any 1–1 function between them is a bijection.

Given a bijection $f\colon X \to Y$, the *inverse function of* $f$, $f^{-1}\colon Y \to X$ is defined by letting $f^{-1}(y) = x$ whenever $f(x) = y$.

A bijection and ... its inverse

*Bijection*s are used for encryption, and their *inverse* are used for decryption.

Some other important functions for cryptography are *one-way* functions and *trapdoor one-way* functions.

A function $f\colon X \to Y$ is a *one-way function* if it's "easy" to compute $f(x)$, but, for *almost* all $y$s it's hard (*computationally infeasible*) to compute an $x$ such that $f(x) = y$.

Note that the definition of a one-way function allows that for some $y$s it may be way to compute an $x$ such that $f(x) = y$, but this should hold only for a *small* number of $y$s; for the rest of them, the computation of such an $x$ should be hard.

A *trapdoor one-way function* is a one-way function with the additional property that given some extra information, the *trapdoor information*, it becomes feasible to find for any given $y \in \mathrm{Im}(Y)$ an $x \in X$ such that $y = f(x)$.

*One-way functions form the basis for public-key cryptography.*

## Examples of One-Way Function

**Example 1:** Let $X = \{1, \ldots, 16\}$ and define $f(x)$ to be the remainder of

$3^x$ when divided by 17. See Figure 4.

| $x$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 3 | 9 | 10 | 13 | 5 | 15 | 11 | 16 |
| | | | | | | | | |
| $x$ | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $f(x)$ | 14 | 8 | 7 | 4 | 12 | 2 | 6 | 1 |

Figure 4: $3^x \pmod{17}$

Given a number $x \in X$, it's relatively easy to find $f(x)$. But, without the table, it is not easy to find a $y$ such that $x = f(y)$ (unless, of course, $x = 3$.) Even when the numbers are much larger, computing $f$ is easy, while computing $f^{-1}$ is much harder.

**Example 2** Let $p = 48,611$ $q = 53,993$, and $n = pq = 2,624,653,723$, Thus $n$ is a product of two odd primes, $p$ and $q$. Let $X = \{1, \ldots, n-1\}$, and $f(x)$ be the remainder of $x^3$ when divided by $n$. E.g., $f(2,489,991) = 1,981,394,214$ since

$2489911^3 = 5881949859 \cdot n + 1981394214$.

The function $f$ here is "easy" to compute (once you have some number theoretic results at your disposal.) However, if $n$ is known, but $p$ and $q$ are not, and both are large, then $f$ is a one-way function. This is an example of a *modular cube root with modulus* $n$, which is a difficult program if the prime factors of $n$ are unknown. However, if the factors of $n$ ($p$ and $q$) are known, then there is an algorithm for computing the cube roots.

In fact, the security of RSA depends on the difficulty of finding the $e^{th}$ roots modulo $n$ where $n$ is a producct of two distinct (large) odd primes.

Thus, $f$ is a trapdoor one-way function, with the prime factors of $n$ being the trapdoor information.

## A Note on One-Way Functions

It remains to be established whether there are any true one-way functions: *No one has yet definitely proved the existence of such functions* under reasonable definitions of "easy" and "hard" to compute.

It therefore follows that the existence of trapdoor one-way functions is also unknown. There are, however, a number of good candidates.

## Permutations and Involutions

Let $S$ be a set of elements.

A *permutation on* $S$ is a bijection from $S$ to itself.

An *involution* on $S$ is a permutation $f$ on $S$ such that $f = f^{-1}$, i.e., for all $x \in S$, $f(f(x)) = x$. E.g., if $S = \{1, 2, 3, 4, 5\}$ and $f(1) = 4$, $f(2) = 2$, $f(3) = 5$, $f(4) = 1$, and $f(5) = 3$, then $f$ is an involution.

# Terminology and Concepts

$\mathcal{A}$ A finite alphabet over which, e.g., messages are composed. Examples are the *binary alphabet* $\mathcal{A} = \{0, 1\}$, or the English alphabet $\mathcal{A} = \{a, ..., z, A, ..., Z\}$. Every alphabet can be encoded by the binary alphabet.

$\mathcal{M}$ A message space. Its elements are (usually finite) strings from the alphabet. Elements of $\mathcal{M}$ are called *plaintext messages*, or, simply, *plaintexts*.

$\mathcal{C}$ is a *ciphertext space*. Its elements are (usually finite) strings from the alphabet that are *not* plaintexts. Elements of $\mathcal{C}$ are called *ciphertexts*.

$\mathcal{K}$ is a key space, whose elements are called keys. Each element $e \in \mathcal{K}$ determines a bijection $E_e$ from $\mathcal{M}$ to $\mathcal{C}$ called encryption transformation. Similarly, each $d \in \mathcal{K}$ determines a bijection $D_d$ from $\mathcal{C}$ to $\mathcal{M}$ called decryption transformation.

**Encryption Scheme** An encryption scheme consists of two sets, $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$ such that for every key $e \in \mathcal{K}$ there is a unique key $d \in \mathcal{K}$ such that $D_d = E_e^{-1}$, i.e., for every message $m$, $D_d(E_e(m)) = m$. Such a pair $(e, d)$ is often called a key pair.

**Cryptoanalysis** The study of mathematical techniques to breaking cryptographic systems.

**Cryptology** The study of cryptography and cryptoanalysis.

**Cryptosystem** The set of cryptographic primitives, usually to achieve confidentiality.