SYSTÈME & RÉSEAUX

1. Rudiments de cryptographie

L'objectif de ce cours est de vous donner un aperçu des techniques de cryptographie qui sont utilisées dans des protocoles comme PGP, ssh ou ssl: chiffrage et déchiffrage, signature, authentification.

1.1. Problèmes.

- Chiffrement / déchiffrement (protocoles symétriques / non symétriques)
- Signature
- Authentification
- Tatouage (watermark)
- Dissimulation

1.2. Un exemple d'algorithme de cryptographie: RSA.

La méthode de cryptographie RSA, codage à clef publique, a été inventée en 1977 par R.-L. Rivest, A. Shamir et L. Adleman *A method for obtaining digital signatures and Public-Key cryptosystems*, communications of the A.C.M., février 1978.

Quelques rappels sur les nombres premiers et les entiers modulos p (Z/pZ).

Exercice 1. Trouver les nombres inversibles dans $Z/5Z$ et $Z/6Z$.

Théorème. *Petit théorème de Fermat:*
Si p est un nombre premier, alors pour tout entier m, alors

$$m^p = m \mod p.$$

Variations:

- Si p est un nombre premier, et si m n'est pas divisible par p, alors
$$m^{p-1} = 1 \mod p.$$
- Si p et q sont deux nombres premiers, et si m est premier avec p et q, alors
$$m^{(p-1)(q-1)} = 1 \mod pq.$$

La méthode RSA est basée sur cette dernière variation. On prend deux grands nombres premiers p et q, et on note n le produit pq. On choisit ensuite un nombre e premier avec $(p-1)(q-1)$. Alors e est inversible modulo $(p-1)(q-1)$. On note d son inverse. On appelle alors:

- clef publique le couple (n, e).
- clef privée le couple (n, d);

L'algorithme fonctionne alors comme suit:

Algorithme 1.1. *[Rivest, Shamir et Adleman]*
Soit $M < n$ un nombre appelé message. Le message chiffré est alors le nombre codé à l'aide de la clef publique

$$C := M^e \mod n.$$

On retrouve alors M à l'aide de la clef privée par

$$M' := C^d \mod n.$$

Exercice. Vérifier que l'on retrouve bien le message d'origine M.

Vérifier que les deux clef jouent des rôles symétriques: si on chiffre avec la clef privée puis l'on déchiffre avec la clef publique, alors on reobtient encore le message d'origine. On dit que le chiffrement est *commutatif*.

Exercice. Connaissant p, q, et e, comment peut-on calculer d ?

La sécurité de ce chiffrement repose sur le fait qu'il est très difficile d'obtenir d sans connaître la factorisation pq de n.

1.2.1. *Implantation (mercredi 3 novembre).* Nous allons maintenant implanter cet algorithme en java (on utilisera un système de calcul formel comme Maple ou MuPAD pour la génération de nombres premiers et la factorisation). Le TP du mercredi 10 décembre sera basé sur cette implantation et sera noté. Il est donc indispensable que vous ayez d'ici là une version fonctionnelle.

Exercice. Nombres premiers en Maple, MuPAD, ...

Avec les commandes Maple ou MuPAD `nextprime`, `prevprime` et `ithprime`, afficher des nombres premiers à 10 et 20 chiffres. Effectuer le produit de deux nombres premiers, puis tenter de factoriser ce résultat avec la commande `ifactor`/`factor`. Que pouvez-vous conclure ? Écrire une fonction qui renvoie un nombre premier "aléatoire" entre 10^n et 10^{n+1}.

Exercice. Implantation en java: complétez le squelette fourni: `http://lapcs.univ-lyon1.fr/~nthiery/DESS/Notes/Exemples/OI-TP1-Nicolas.Thiery-1.0.tar.gz`.

Note: on demande d'implanter des fonctions pour calculer puissances et inverses modulo m, même si elles sont en fait déjà disponibles dans BigInteger.

1.2.2. *Utilisation de RSA (mercredi 10 novembre).*

Exercice 2. On va expérimenter en taille réelle pour voir comment on peut arriver à échanger des données confidentielles par dessus un media non sûr.

(1) Le répertoire /home/Media sera le média non sûr par dessus lequel vous allez communiquer (tout le monde peut y lire et écrire librement). Pour entrer dans le répertoire, taper
`cd /home/Media`
(2) Créez une paire clef publique/clef privée, et mettez la clef publique (c'est-à-dire n et e) dans un fichier `Prenom.Nom.clef-publique`, par exemple en utilisant `emacs`.
(3) Pour écrire un message à quelqu'un, écrire le message chiffré dans un fichier avec un nom dans le genre `Prenom1.Nom1-pour-Prenom2.Nom2-numero`, ou 1 est l'expéditeur, 2 est le destinataire, et `numero` est le numéro du message. Typiquement: `Nicolas.Thiery-pour-Florent.Hivert-4`.
(4) Pour visualiser un message, vous pourrez par exemple utiliser
`cat Nicolas.Thiery-pour-Florent.Hivert-4`
(5) Le copier-coller sera très utile pour toutes ces manoeuvres! De même que l'utilisation de la touche TAB pour la complétion automatique des noms de fichiers.

(6) Évidement, vous êtes plus qu'encouragés à essayer d'intercepter les messages de vos petits camarades :-)

Pour mettre un peu de piment, on va faire un petit championnat:

- Chaque message envoyé de manière sûre donne un point à l'expéditeur et au destinataire;
- Chaque message intercepté et déchiffré par une tierce personne lui donne trois points;
- Celui qui marque le plus grand nombre de points a gagné.

Dans le répertoire, il y a déjà ma clef publique, un message que j'ai chiffré avec ma clef privée à l'intention de tout le monde, la clef publique de Toto et un message chiffré à l'intention de Toto. Toto a une clef privée que vous ne connaissez pas, votre objectif est de déchiffrer ce message.

Le TP sera noté sur 10. Il sera à rendre comme d'habitude sous la forme d'une archive contenant le code java, vos clefs publiques et privées, le message chiffré à mon intention, et le contenu déchiffré du message pour tout le monde et du message pour Toto.

1.3. **Applications.**

Exercice. Applications de RSA

Décrivez comment on peut utiliser RSA pour résoudre les problèmes suivants:

(1) Chiffrement
(2) Signature
(3) Authentification

À votre avis, quelles sont les difficultés qui peuvent être rencontrées dans la mise en oeuvre de RSA ?

(1) Garantie de qualité du générateur aléatoire de nombres premiers
(2) Garantie que n ne soit pas factorisable
(3) Compromis vitesse de calcul / niveau de sécurité
(4) Gestion du risque inhérent de tomber sur un message non premier avec $(p-1)(q-1)$
(5) Gestion des clefs publiques
(6) Protection des clefs privées
(7) Standardisation
(8) Difficultés légales
(9) Difficultés avec les brevets

Exercice. Vous devez envoyer à un correspondant une petite mallette pouvant être fermée par un ou plusieurs cadenas, et contenant des documents ultra-confidentiels mais non urgents. Les seuls moyens à votre disposition pour communiquer avec votre correspondant sont la poste et le téléphone, qui sont notoirement non sûrs. Comment procéder ?

1.4. Principe de fonctionnement de PGP / GNU PG.

(1) Certification de clefs, réseau de confiance, serveurs de clefs, ...

1.5. Principe de fonctionnement de ssh.

(1) Authentification avec ou sans échange de mot de passe
(2) Échange de clef de session + chiffrement symétrique

1.6. Références.

(1) Site web de GNU PG `http://www.gnupg.org/`
(2) GNU Privacy Handbook `http://www.gnupg.org/gph/en/manual.html` Ce manuel (en anglais) contient une excellente introduction à l'utilisation d'outils de cryptographie pour protéger sa vie privée.
(3) Documentation de GNU PG `http://www.gnupg.org/howtos/fr/` Version abrégée et traduite du précédent.
(4) Site web d'openssh `http://www.openssh.org`
(5) Une introduction a la cryptographie `http://www.di.ens.fr/~wwwgrecc/Recherche/Crypto/intro/intro.html`
(6) Une discussion sur la taille des cles de chiffrement `http://www.di.ens.fr/~pornin/faq-cle.html`

2. Introduction au fonctionnement des réseaux

Je vous conseille de lire le document suivant `http://www.eleves.ens.fr:8080/tuteurs/hublot/rezo.ps.gz`. De manière générale, je recommande chaleureusement le site des tuteurs `http://www.eleves.ens.fr/tuteurs/`, qui contient de très bons documents d'introduction.

Exercice 3. Utilisez la commande `traceroute machine` depuis votre connexion ssh sur lapcs.univ-lyon1.fr pour explorer la topologie du réseau aux alentours de la fac. Vous pourrez par exemple commencer par `traceroute www.google.fr`, ou toute autre machine de votre choix. Au fur et à mesure, dessinez le réseau sur le tableau noir. Préciser au fur et à mesure les noms, adresse IP, et si possible adresses de sous-réseaux.

Expérimentez aussi avec les commandes ping, host, route.

Exercice 4. Quelques ordres de grandeurs:

Évaluer le nombre d'octets nécessaires pour stocker:

(1) Un nombre entier sur 5 chiffres;
(2) Un nombre flottant;
(3) Un caractère;
(4) Une page de texte;
(5) La bible;
(6) Une photo;
(7) Une heure de son;
(8) Une heure de vidéo;

En déduire, à la louche, la capacité d'un CDROM et d'un DVD.

Évaluer la quantité d'images que l'on pourrait stocker dans une camionnette.

Évaluer le débit nécessaire pour transmettre:

(1) Du texte tapé à la main (talk, IRC, et autres messageries instantanées);
(2) Du son;

(3) De la vidéo.

Évaluer le temps de latence maximal pour transmettre:

(1) Du texte tapé à la main;
(2) Du son;
(3) De la vidéo.

Votre machine chez vous est connectée à internet par le téléphone soit par une connexion normale (~50 kb/s) ou par ADSL (~512 kb/s). Évaluer dans les deux cas quel est le moyen le plus rapide de rapatrier le contenu d'un DVD disponible sur internet.

3. Calculs scientifiques et réseaux

Objectif: Comprendre comment on peut utiliser le réseau pour faire des calculs scientifiques

Problème. Vous avez un super nouvel algorithme pour, mettons, faire une certaine analyse statistique d'un jeu de données et vous voulez que le plus grand nombre de personnes puissent l'utiliser.

Jusqu'à la fin de ce cours, vous allez découvrir différentes possibilités pour faire cela, en implantant des petits jeux en réseaux.

3.1. **Communications entre processus par tuyaux et prises (pipes & sockets, ...)** Un réseau c'est a priori compliqué. Le but est de fournir une interface qui permette à des programmes (processus) de communiquer simplement entre eux, en faisant abstraction de la complexité du réseau en dessous.

Rien de sorcier, c'est comme le téléphone:

- Le numéro de téléphone est remplacé par un nom de machine
- Comme il peut y avoir plusieurs programmes sur une machine (plusieurs personnes pour un même numéro de téléphone) on rajoute un numéro de port (numéro de poste).

Voilà comment ça se passe:

(1) Le programme A sur la machine alpha demande au système d'attendre les connections sur le port 6083.
(2) Le programme B sur la machine beta demande au système de se connecter sur la machine alpha sur le port 6083.
(3) Le programme A accepte la connexion et décroche; la connexion est établie.
(4) Les programmes A et B ont à leur disposition deux tuyaux leur permettant de communiquer.
(5) Quand ils ont terminé ils coupent la communication.

3.2. **Exemples de protocoles réseaux: http (web), smtp (mail), nntp (Usenet), ...** Ce sont des protocoles client-serveur typiques:

(1) A est un serveur web (apache, ...) qui tourne sur le port www (80) de alpha
(2) B est un navigateur web (netscape, ...) qui se connecte sur alpha
(3) A accepte la connexion, et démarre un nouveau processus A' en redirigeant les entrées et sorties standard de celui-ci vers le tuyau.
(4) A reprends l'attente de nouvelles connections
(5) A' envoie (éventuellement) un message d'accueil
(6) B demande une page web

(7) A' renvoie la page web puis raccroche

Exemple. Récupérer un fichier à la main sur un serveur web:

```
icare> telnet lapcs.univ-lyon1.fr www
GET /index.html
```

Version un peu plus avancée:

```
icare> telnet lapcs.univ-lyon1.fr www
GET /index.html HTTP/1.0
```

Exemple. Envoyer un mail à la main:

```
icare> telnet localhost smtp
HELO univ-lyon1.fr
MAIL FROM: le.pere.noel@antartique.fr
RCPT TO: nthiery@localhost
DATA
Voilà plein de cadeaux!
                              Le père noël
.
QUIT
```

Cette méthode permet d'envoyer des faux mails.

Elle ne marche pas toujours, car il y a des protections.

De toutes les façons, IL NE FAUT PAS EN ABUSER !

Si un faux mail porte préjudice à quelqu'un, celui-ci peut engager des poursuites judiciaires, comme pour tout usage de faux documents.

Exercice. Lire les news à la main:

Essayez les commandes suivantes:

```
lapcs> telnet news.univ-lyon1.fr nntp
HELP
LIST
GROUP fr.comp.lang.php
BODY 15090
HEAD 15090
ARTICLE 15090
NEXT
HEAD
BODY
ARTICLE
```

Exercice. Utiliser l'exercice précédent et les commandes UNIX que vous avez vues pour récupérer la liste de tous les groupes de discussion francais (fr.quelquechose).

Indication: telnet est une application UNIX comme les autres; vous pouvez donc rediriger à volonté son entrée et sa sortie standard.

Ne regardez pas la correction ci-dessous avant d'avoir vraiment essayé par vous-même!

Correction. Version minimale:

```
echo list | telnet news.univ-lyon1.fr nntp | grep fr
```

Version avancée plus polie avec le serveur et garantissant que le fr est en début de ligne:

```
(echo list; echo quit) | telnet news.univ-lyon1.fr nntp | grep '^fr\.'
```

Expliquez ce qu'il ce passe lorsque l'on exécute ces commande!

3.3. **Comment cela marche les tuyaux ?** On utilise des niveaux d'abstraction successifs, chaque couche apportant une amélioration à la "qualité de service". Dans l'implantation de chaque couche, on a pas besoin de se préoccuper des détails d'implantation des couches inférieures.

(1) Envoi d'un signal entre deux machines: couche physique (câble Ethernet, ligne de téléphone (normal/ADSL), fibre optique, faisceau lumineux, radio (faisceaux herziens, téléphone portable, réseau local sans fil), télégraphe, pigeon voyageur...)
(2) Envoi d'un paquet entre deux machines directement connectées: Ethernet, PPP, PLIP, ...
(3) Envoi d'un paquet avec routage entre machines sur des réseaux différents: IP
(4) Envoi de suites de paquets avec garantie de qualité: TCP
(5) Tuyaux et prises (socket)
(6) Tuyaux sécurisés (authentification, chiffrement): ssl/ssh
(7) Protocoles réseaux (http, smtp, nntp, nfs, samba, netware, ntp, ...)
(8) Interfaces utilisateur (navigateur web, logiciel de mail, lecteur de news, ...)

Toutes ces couches font l'objet de spécifications précises dans des standards. La plupart du temps sous forme de documents RFC (Request For Comments). Le développement d'Internet a été essentiellement permis par le fait que ces standards soient *ouverts*.

3.4. **TP: les prises et tuyaux en java.** La bibliothèque standard de java contient toutes les fonctions nécessaires pour utiliser des prises et tuyaux. Vous trouverez deux exemples d'utilisation dans l'archive suivante:`http://lapcs.univ-lyon1.fr/~nthiery/DESS/Notes/Exemples/SR-TP3-Nicolas.Thiery-1.0.tar.gz`

Ce TP ne sera pas noté. Il est à faire pour le mercredi 29 janvier.

Exercice 5. Écrivez une application java `Aspirateur` qui prends comme argument sur la ligne de commande l'URL d'un document sur le web, se connecte sur le serveur web correspondant et rappatrie le document dans un fichier. Par exemple, si on tape

```
java Aspirateur http://lapcs.univ-lyon1.fr/~nthiery/DESS/index.html
```

l'application doit récupérer le fichier `index.html` et le sauvegarder dans le répertoire courant.

Exercice 6. Écrivez un serveur qui calcule $n!$.

Description du protocole:

(1) Le serveur attends la connection d'un client.
(2) Le client se connecte.
(3) Le client envoie au serveur une ligne contenant un nombre n.
(4) Le serveur lui renvoie une line contenant $n!$.
(5) Boucle en 3.

De plus:

- Pour se déconnecter, le client envoie une ligne contennant la commande `quit`.
- Si le client envoie n'importe quoi (nombre négatif, commande inconnue, ...), le serveur lui renvoie une ligne avec un message d'erreur.

Exercice 7. Écrivez deux applications (un client et un serveur) qui permettent à deux personnes sur des machines différentes de dialoguer en s'envoyant des lignes de texte à tour de rôle. Mettez-vous d'accord entre groupes pour que tous vos programmes soient compatibles entre eux!

3.5. **Calculs scientifiques et réseaux.**

Problème. En quoi les réseaux peuvent-ils être utiles pour les calculs scientifiques ?

- Distribution rapide de logiciels
- Distribution simplifiée de logiciels complexes à installer
  Versions de démonstration
- Lancement de calculs sur plusieurs machines
- Calcul parallèle (PVM, MPI, clusters, machines spécialisées)
- Calcul massivement parallèle (seti@home)
- Mise à disposition de puissance de calcul et de logiciels
  (centre de calcul medicis.polytechnique.fr)
- Bases de données (encyclopédie des séquences d'entiers)
- Documents scientifiques interactifs, éducation en ligne

Problème. Quelles solutions techniques?

- Côté client (on fournit le logiciel qui est exécuté sur le client):
  (1) Programme autonome
  (2) Bibliothèque pour un système existant
  (3) Bibliothèque précompilée (C, C++, fortran, java, ...) à lier avec un autre programme
  (4) Code exécuté dans le navigateur web: HTML+Javascript / Applet java
  (5) Distribution: source / binaire ? licence ?
- Côté serveur (le logiciel est exécuté sur le serveur):
  (1) Connexion par ssh sur le serveur, et travail à distance avec les logiciels qui y sont installés (medicis.polytechnique.fr)
  (2) Interface web (linbox, encyclopédie des séquences d'entiers):

(a) "scripts" CGI (programme appelé par le serveur web), souvent en perl
(b) PHP/ASP + bases de données (code exécuté directement par le serveur web)

(3) Interface avec protocole dédié (serveurs de calcul FGb+RS)

Problèmes principaux: sécurité, simplicité d'emploi, fiabilité, efficacité, pérennité

Problem 3.1. Quels sont les avantages et inconvénients de chacune d'entre elles? Pour chacune des applications ci-dessus, quelles sont les solutions raisonnables ?

3.6. **Projet: jeu de Hex en réseau.** L'objectif du projet est d'améliorer le jeu de Hex que vous avez programmé en utilisant le réseau.

Exercice 8. On suppose que deux programmes de jeu de dame sont connectés l'un à l'autre par un tuyau. Il va falloir décider comment ces deux programmes vont utiliser ce tuyau pour transmettre les informations nécessaires pour jouer l'un contre l'autre. Commencons par un exemple: voilà un dialogue typique entre deux humains jouant aux dames:

(1) Joueur 1: Bonjour, tu joues aux dames avec moi ?
(2) Joueur 2: Ok. je joue C3-D2
(3) Joueur 1: je joue F4-E5
(4) Joueur 2: je joue C5-E5
(5) Joueur 1: nan, tu n'as pas le droit; essaye encore.
(6) Joueur 2: je joue C5-E5
(7) ...
(8) Joueur 2: je joue F4-H6, t'as perdu!
(9) Joueur 1: zut. Je joue plus.
(10) Joueur 2: mauvais joueur; tant pis, à bientôt

Un programme ne serait probablement pas capable de suivre un tel dialogue. Il y a des ambiguités, des bouts inutiles, des bouts sous-entendus. Le but de l'exercice est de formaliser ce type de dialogue pour qu'il puisse avoir lieu entre deux programmes; c'est-à-dire de mettre au point un protocole permettant à ces programmes d'échanger leurs coups pour mener à bien leur partie.

Exercice 9. Joueur humain contre machine distante.

(1) Écrire une application java qui implante ce protocole via son entrée et sa sortie standard. Ainsi, un joueur connaissant le protocole peut jouer directement avec lui en le lançant à la main. Vous pouvez écrire plusieurs variantes, avec des stratégies de jeu différentes.
(2) Modifier le programme précédent pour en faire un serveur qui attends sur un port donné d'une machine distante.
(3) Améliorer ce serveur pour qu'il gère plusieurs coups en même temps.
(4) Modifier l'interface utilisateur pour qu'elle se connecte sur le serveur précédent pour permette à l'utilisateur de jouer contre la machine. Pour cela, vous pouvez créer une nouvelle classe "RemotePlayer"; lorsqu'un objet `o` de cette classe est créé, il ouvre une connexion avec le serveur distant. Ensuite, lorsque l'on appelle la méthode nextMove de cet objet, celle-ci envoie les informations au serveur, et récupère le mouvement suivant.

Exercice 10. Joueurs humains à distance.

Écrire un serveur "arbitre" qui attends que les interfaces des deux joueurs (ou plus) se connectent dessus, et gère leur dialogue.

Optionnel: étendre ce serveur pour qu'il gère plusieurs parties en même temps.

TODO: faire utiliser les threads

3.7. **Les Threads, ou processus légers.** Références:

- Tutorial java de SUN sur les processus légers `http://java.sun.com/docs/books/tutorial/essential/threads/`
- Des notes de cours en français sur les processus légers `[http://javahttp://cedric.cnam.fr/~farinone/CCAM/threads.pdf`

Exercice 11. Écrire une version du programme joueur humain contre joueur automatique sur la machine locale pour que le calcul du prochain coup ait lieu dans un autre processus léger. Pour cela, vous pouvez créer une nouvelle classe "ThreadPlayer" dont la méthode nextMove créé un nouveau processus léger; ce processus léger à son tour créé un objet de la classe GreedyPlayer (ou de n'importe quel joueur automatique), et appelle la méthode nextMove de cet objet.

Exercice 12. Parallélisme: Écrire un joueur automatique qui recherche le meilleur coup en explorant récursivement l'arbre des possibilités, en parallélisant le calcul sur plusieurs machines.

Une architecture possible:

(1) On a une liste de machines disponibles.
(2) Sur chaque machine tourne un serveur qui attends des connexions. Lorsqu'il en reçoit une, il lance un sous-processus (*thread*) qui traite la connexion (recevoir une requête, calculer une réponse, la renvoyer et se déconnecter). Pour limiter les dégats, le serveur peut tenir à jour le nombre de sous-processus lancés, et se déclarer «indisponible» s'il y en a trop.
(3) Une requête consiste en la description d'un échiquier, d'une couleur (est-ce aux noirs ou aux blancs de jouer) et d'une profondeur p. Le programme fait une recherche dans l'arbre des possibilités et calcule le meilleur coup pour le joueur considéré, c'est-à-dire le coup qui maximise l'évaluation e à profondeur p. La réponse est composée du meilleur coup et de l'évaluation e. Bien entendu ce problème est à traiter récursivement: on parcoure la liste des coups possibles; pour chacun d'entre eux, on calcule la meilleure contre-attaque pour le joueur adverse, et l'évaluation correspondante; enfin on choisi le coup qui minimise la contre-attaque.
(4) Chaque programme connaît la liste des machines. S'il le souhaite, il peut décider en cours de calcul de déléguer une partie du travail sur une autre machine. Pour cela, il peut rechercher un autre serveur disponible (s'il y en a un), et lui donner à résoudre une des sous-branches.

3.8. **Références.** Exemples de serveurs de calcul sur le web:

- L'encyclopedie electronique des suites d'entiers (Sloane) `http://www.research.att.com/~njas/sequences/`
- Serveur de calcul d'algebre lineaire exacte `linalg.org`

Documents:

- Usenet (protocole nntp) `http://ecole.eu.org/doc/usenet/index.html`
- Les services de base d'internet (ftp http mail, nntp) `http://ecole.eu.org/doc/cours4/internet2/`

- Supports de cours de l'Ecole Ouverte de l'Internet `http://ecole.eu.org/doc/supports.html`
- Documents sur l'utilisation d'UNIX sur Linux-France `http://www.linux-france.org/`
- Documents sur l'utilisation d'UNIX, sur le site des tuteurs `http://www.eleves.ens.fr:/tuteurs/`
- Un tutorial pour javascript `http://www.pageresource.com/jscript/`
- Un tutorial java `http://javaboutique.internet.com/tutorials/Step/Chapter1/index.html`
- Tutorial sur les tuyaux (sockets) et java `http://www.javaworld.com/jw-12-1996/jw-12-sockets.html`
- Les exemples présentés en cours `http://www.lapcs.univ-lyon1.fr/~nthiery/DESS/Notes/Exemples/`