



情報セキュリティ

第12回：2008年6月27日(金)



本日学ぶこと

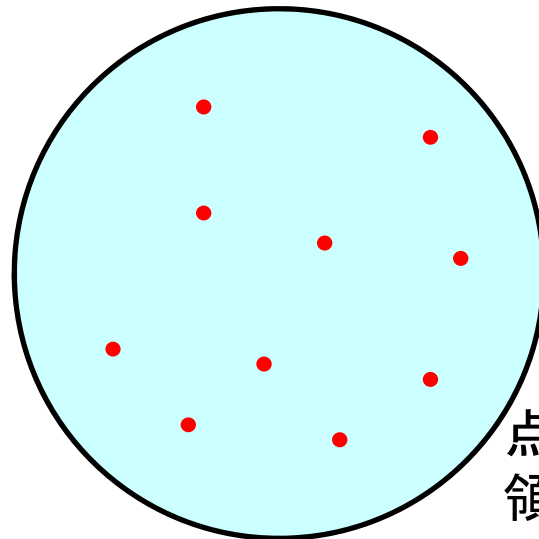
- 暗号の基礎
 - 計算理論
 - 問題とは／計算モデル／オーダ／多項式時間
 - プロトコル
 - プロトコルの諸概念／鍵配布プロトコル(およびその攻撃)／ゼロ知識対話証明
- 現代暗号の安全性を支えているのは「 $P \neq NP$ 」
- 基礎となる暗号技術が安全であっても、それをもとにした暗号プロトコルが安全であるとは限らない

計算理論

- ハードウェア・ソフトウェアの進歩や, アルゴリズムの発明により, それまで解けなかった問題が解けるようになる?
 - 「問題を解く」とは何をすることか見直す必要がある.
- 計算理論に関連するその他の問題意識
 - 乱数とは?
 - P=NP問題って?
 - なぜ $x=1,2,\dots$ と解の候補を試すのが「効率が悪い」のか?

問題と個別問題

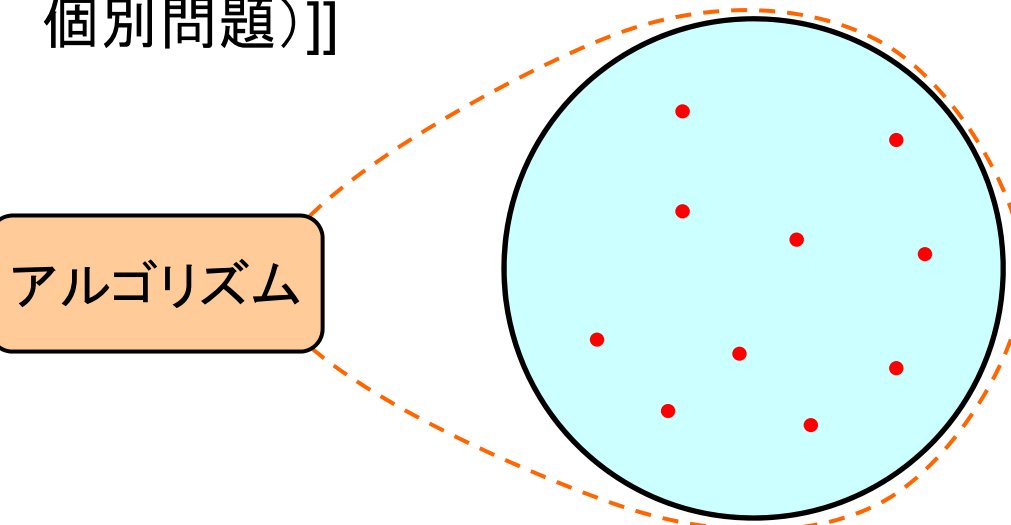
- 素因数分解の個別問題: 35を素因数分解せよ
 - 桁数が大きくても, 事前に答えを知っていれば, その答えを書き写すだけで「解ける」
- 素因数分解問題: 正整数 n が与えられたときに, その素因数の一つを求めよ
- 計算理論での「問題」は, 無限個の「個別問題」を含む.



点...個別問題
領域...問題

問題を解くとは

- 個別問題が解ける: あるアルゴリズムを適用(実行)すると, 有限時間で停止し, 正しい解を出す
- 問題が解ける(決定可能): アルゴリズムが存在して, 問題に属するすべての個別問題が解ける
 - ○ \exists アルゴリズム [\forall 個別問題 \in 問題 [解ける(アルゴリズム, 個別問題)]]
 - × \forall 個別問題 \in 問題 [\exists アルゴリズム [解ける(アルゴリズム, 個別問題)]]

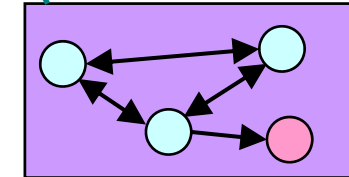


計算モデル...問題を解く「計算機」の抽象化

■ チューリング機械 (Turing Machine)

- 1本の記録用テープを持つ...メモリに相当
- 一つの状態を保存できる...レジスタに相当
- テープのある位置を参照している...アドレスに相当
- 1ステップの計算により, ある時点から, 別の時点に遷移する.
- 状態の遷移の仕方はプログラム,
テープの初期状態は入力に相当
- 「終了状態」と呼ばれる特別な状態に到達すれば, 停止する.

... 0 0 1 1 0 1 0 ...

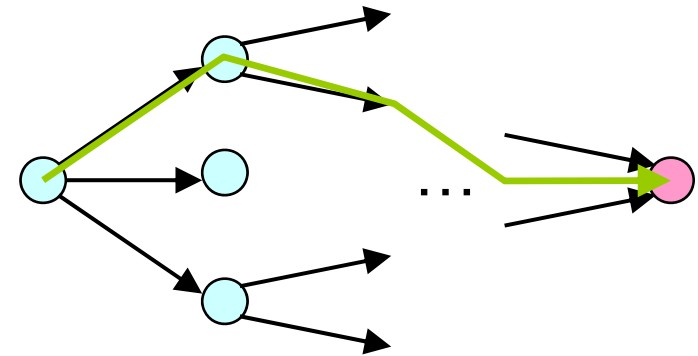


■ 現在のコンピュータとどう違う？

- コンピュータで可能なあらゆる処理をコード化可能
- 無限サイズのメモリを持っている

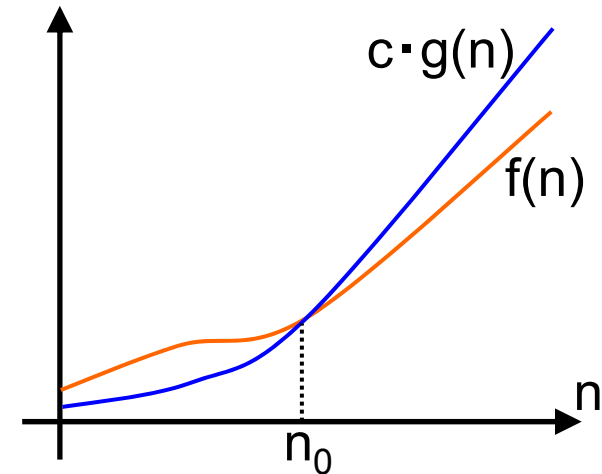
計算モデルの種類

- **決定性** (Deterministic) チューリング機械
 - 各時点に対して, 次の時点はちょうど1個
- **非決定性** (Non-Deterministic) チューリング機械
 - 次の時点の候補が複数あってもよく, その中で都合のいいものを選ぶ
- **確率** (Probabilistic) チューリング機械
 - 次の時点の候補が複数あってもよく, 確率に基づいてどれかを選ぶ
- 性能は?
 - 「解く」ということなら, どれも同じ
 - 「効率よく解く」となると, 違いがあると信じられている



計算量とオーダ記法

- 実行時間(時間計算量)を決める要素
 - アルゴリズム...重要
 - 入力データ...致し方ない
 - 実行環境...重要ではない
- サイズが n の入力データで実行して $f(n)$ の時間がかかるとき,
 $\exists g(x), c, n_0 [\forall n > n_0 [f(n) < c \cdot g(n)]]$ ならば,
時間計算量は $O(g(n))$ ($g(n)$ のオーダ)という.
- 例
 - $f(n) = 5n^3 - 2n^2 + 18 \Rightarrow O(n^3)$
 - $f(n) = 2^n + 100n^{200} \Rightarrow O(2^n)$
 - $f(n) = 10000n \Rightarrow O(n)$
 - $f(n) = 137 \Rightarrow O(1)$



$g(n)$ には簡潔で
タイトな関数を選ぶ

オーダーに関して知っておくこと

- アルゴリズムに関するオーダーは、注文や順番という意味ではなく、「百万のオーダー」というような使い方でもなく、ビッグ・オー記法(ランダウの記号)を用いて表すものを言う
 - たいていは時間計算量. たまに空間計算量(メモリサイズ)も
- 一番次数の高いもの以外, それと係数は無視
 - $f(n)=5n^3-2n^2+18 \Rightarrow O(n^3)$
- 構造化プログラミングでボトムアップに評価できる
 - 順次と分岐は, その中の最大を選ぶ
 - 反復は, 掛け算
- オーダーの「=」は, 数学の「=」ではない
 - $f(n) = 2n^2, g(n) = n^2 + n + 20$ のとき,
 $f(n) = O(n^2), g(n) = O(n^2)$ (fとgは同じオーダー)だが $f(n) \neq g(n)$

オーダーで効率を比較

- ソートアルゴリズムのオーダーは...

- 入力サイズは, ソート対象の要素数
- バブルソート: 平均時で $O(n^2)$
- 選択ソート: 平均時で $O(n^2)$
- クイックソート: 平均時で $O(n \log n)$, 最悪時で $O(n^2)$
- 基数ソート: データが入力サイズに依存しなければ, $O(n)$

2要素の
比較回数

「2要素の比較」をしない

- オーダーの大小関係

- $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < \dots < O(2^n) < O(n!)$

多項式時間

- 処理時間が、適当な定数 k を用いて $O(n^k)$ となるアルゴリズムを、多項式時間 (Polynomial-Time) アルゴリズムという。
 - 暗号理論では「効率のよいアルゴリズム」
- 総当り法で離散対数問題を解くのは、多項式時間アルゴリズムではない。
 - $a^b \bmod p = r$ から b を求める
 - $b=1,2,\dots$ と解の候補を試すと、解が出るまでのべき剰余計算回数の期待値は、おおよそ $p/2$. すなわち p に比例する.
 - 離散対数問題の入力サイズは、 p のビット長 $|p| \doteq \log_2 p$ によって決まる.
 - よって、時間計算量は $O(p) = O(2^{|p|})$ であり、多項式時間ではない.

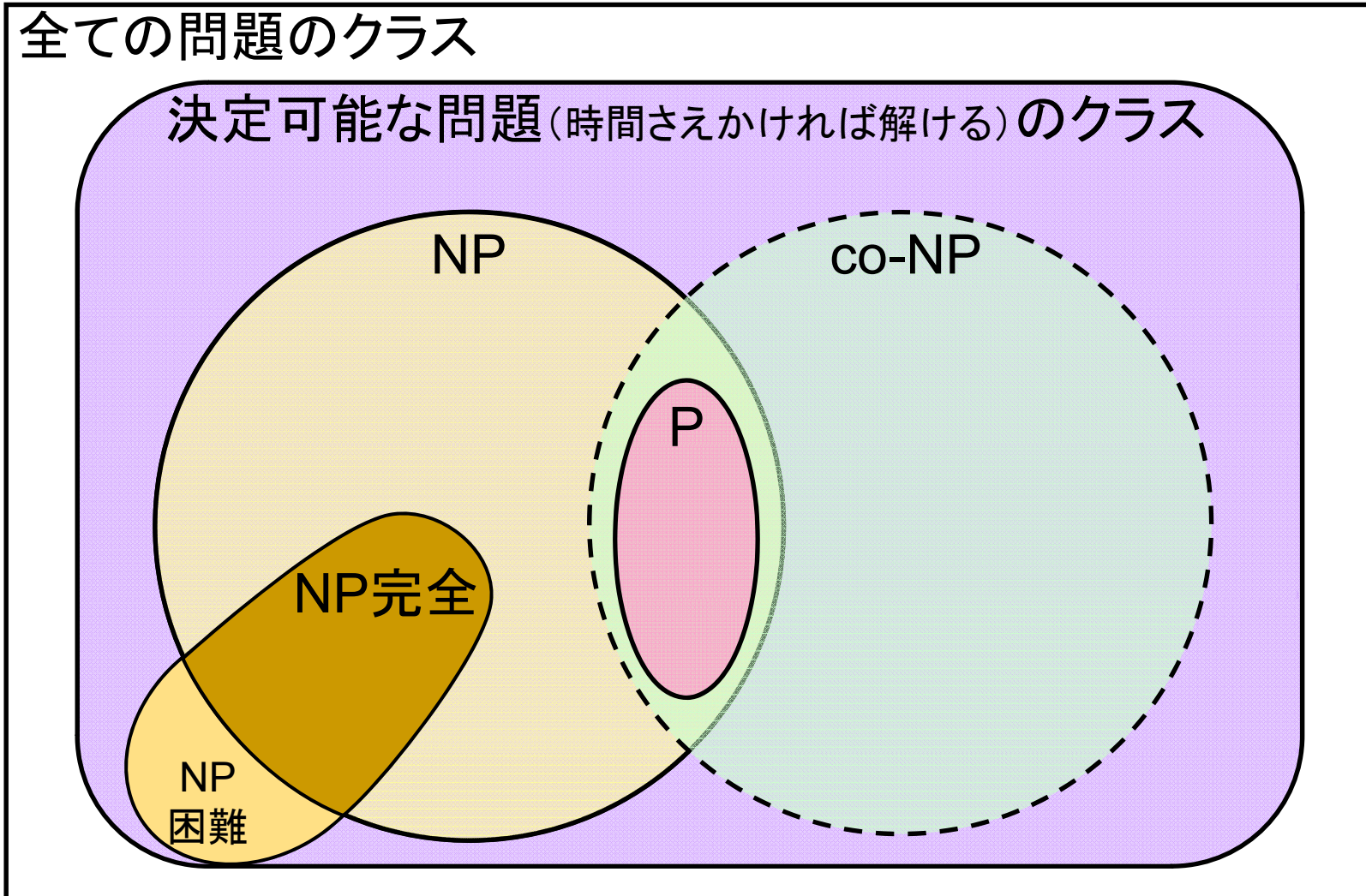
PとNP

- **P**: 決定性チューリング機械を用いて多項式時間で計算できる問題の集合
- **NP**: 非決定性チューリング機械を用いて多項式時間で計算できる問題の集合
- **P=NP問題**: $P=NP$ か, $P \neq NP$ (P が NP に真に含まれる)か
 - 多くの計算機科学者は $P \neq NP$ であると信じているが, まだその証明はなされていない.
 - 素因数分解や離散対数問題などは NP に属する.
もし $P=NP$ ならこれらは P に属することになり, 暗号理論の常識が崩れてしまう.
- **NP完全問題**: NP に属するどの問題も, 多項式時間の変換により帰着できるような問題
 - NP 完全問題の一つでも P に属すれば, $P=NP$ と言える

解けない問題

- チューリング機械で解けない問題(決定不能)
 - 例: 停止性問題... チューリング機械のプログラムコードと入力を与えられたとき, そのチューリング機械がその入力で計算をしたら停止するか?
- 非決定性では効率よく解けるが, 決定性では効率よく解けない(と思われている)問題
 - 素因数分解, 離散対数など

問題の分類



点...問題, 領域...問題のクラス

計算理論のまとめ

- 「問題」を見つけたら
 - 無限個の個別問題が分かるよう、定式化できるか？
 - そもそも解ける(アルゴリズムを考えて、任意の個別問題がそれで解ける)か？
 - 解けないとき、何らかの制限により、解けるようになるか？
 - 解けるとき、推測なしで(決定性で)効率よく解けるか？
 - 解けないとき、NP完全か？あるいは、何らかの制限により、効率よく解けるようになるか？
 - どれくらいの効率か、オーダで表現できないか？
 - 推測(乱数生成)を使って解く場合、成功確率は？ 試行回数はいくらぐらいあればいい？

プロトコルとは

- 2者以上の参加者が関係し、ある課題を達成するための一連の手順のこと。
 - 一連の手順(シーケンス)...前のステップが終わっていないのに次のステップを始めてはならない.
 - 2者以上の参加者が関係...「ケーキを焼く」はプロトコルではないが、「だれかに食べてもらう」まで含めればプロトコルになる.
 - ある課題を達成...これがないければ時間の無駄

プロトコルと関連する話題

- アルゴリズムも、複数エンティティで実行するなら「プロトコル」と言える ⇒分散コンピューティング
- 暗号化した通信に限らず、暗号技術(素因数分解問題や離散対数問題の困難性なども)を用いた通信方法は「暗号プロトコル」という
 - ハッシュ値の照合や、デジタル署名も、暗号プロトコルになり得る.
 - PKIにおける鍵の入手, Diffie-Hellman鍵交換, SSL/TLSやSSHのハンドシェイクプロトコルも該当する.

プロトコルの設計・実行

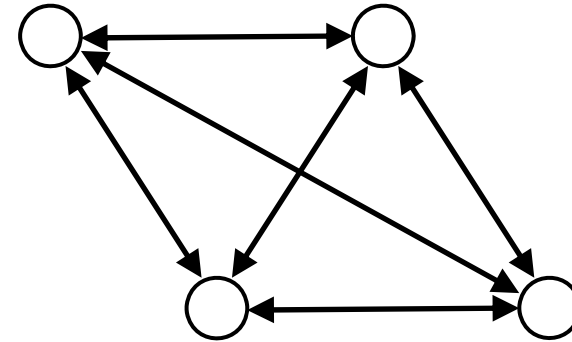
- 適切に設計されたプロトコルは、デッドロックなどの通信上の障害を起こさない。
 - 双方が相手の受信待ち状態になってはいけない！
- 安全に設計された暗号プロトコルは、悪意のある送信を何らかの方法で検出する。
 - 適切な情報を持たない者を誤って認証してはいけない！

暗号プロトコルの具体例

- 鍵配布センターを利用したセッション鍵共有
- Feige-Fiat-Shamirの認証プロトコル

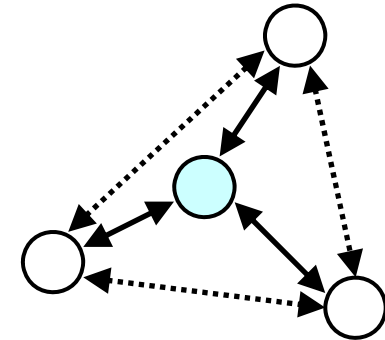
対称暗号をどう使う？（復習）

- これからの議論の仮定：使用する対称暗号は安全である
- 対称暗号は、暗号化の鍵＝復号の鍵
- 鍵はいくつ必要？
 - n 人ならそれぞれ $n-1$ 個、全体で $n(n-1)/2$ 個



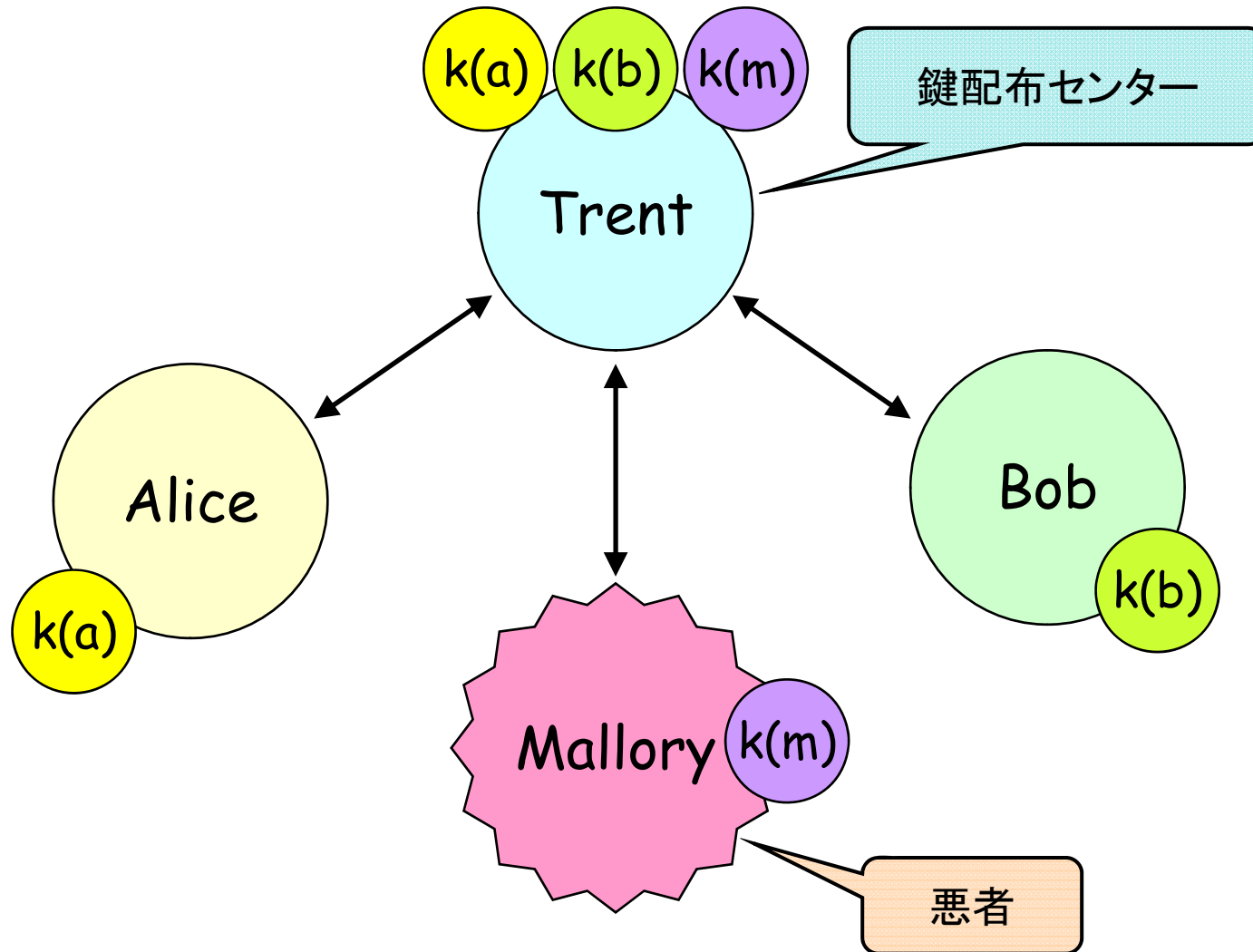
鍵配布センター

- **鍵配布センター**を設置し, 各利用者は, ことと鍵を共有する.
 - 鍵の数は, n 人ならそれぞれ1個, 全体で n 個
- 鍵配布センターは, 不正や鍵の漏洩などをしないものとする.
 - 信頼される第三者 (Trusted Third Party) と呼ばれる.
 - 利用者の中には, 他人の秘密情報を知りたい「悪者」がいるかもしれない.
- 利用者間の通信には, その場限りの鍵 (**セッション鍵**) を生成して使ってもらう.
 - セッション鍵は, 乱数を用いて生成し, 各利用者の鍵や, これまで生成したセッション鍵に依存しないものにする.

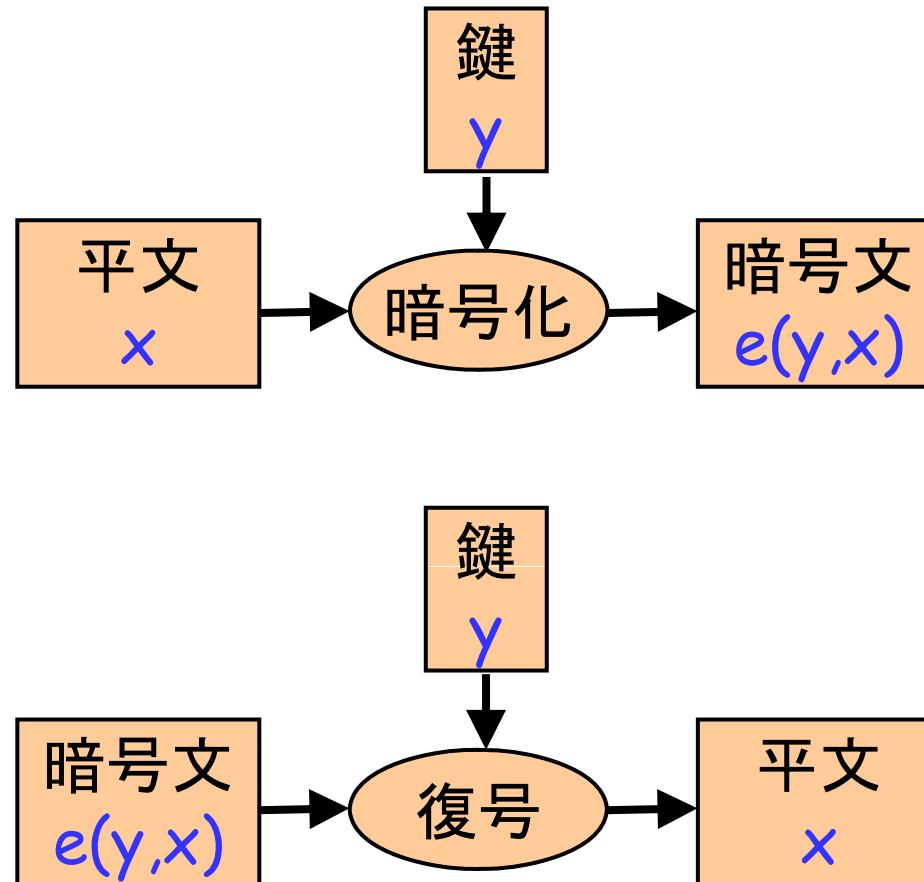


でも, どうすればセッション鍵を
当事者間(のみ)で共有できる?

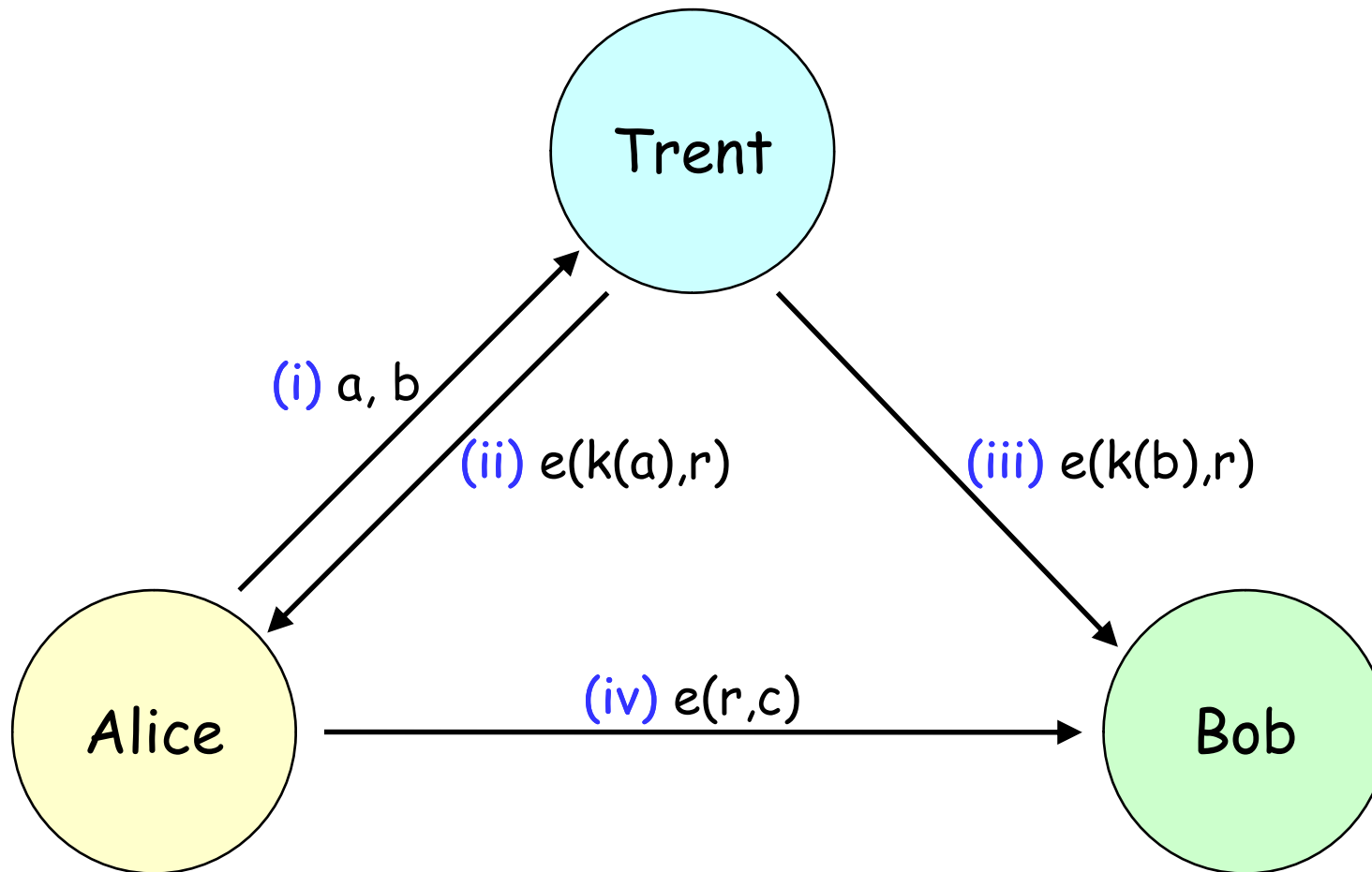
準備(1): 実行環境



準備(2): 暗号化と復号の記法



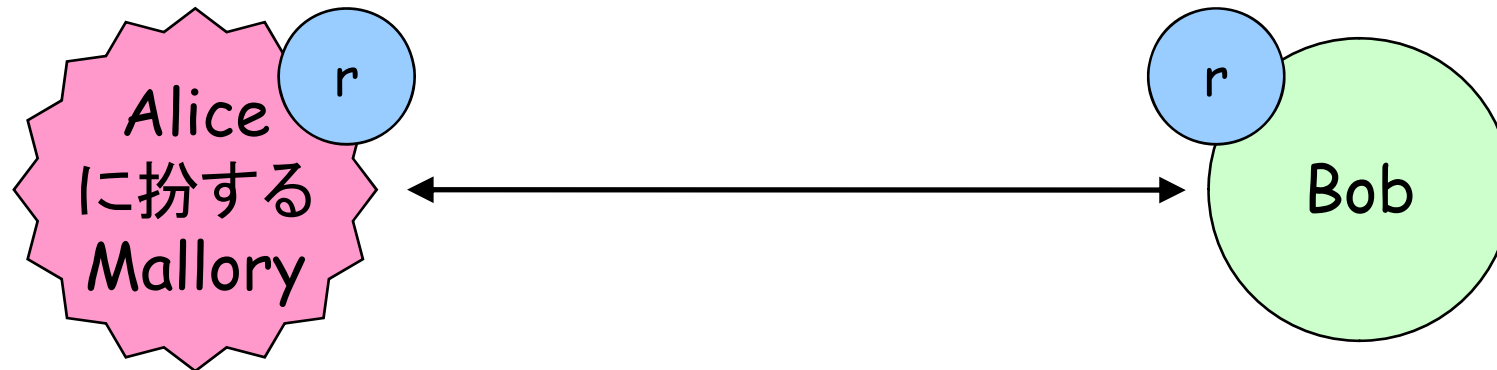
セッション鍵の配布案



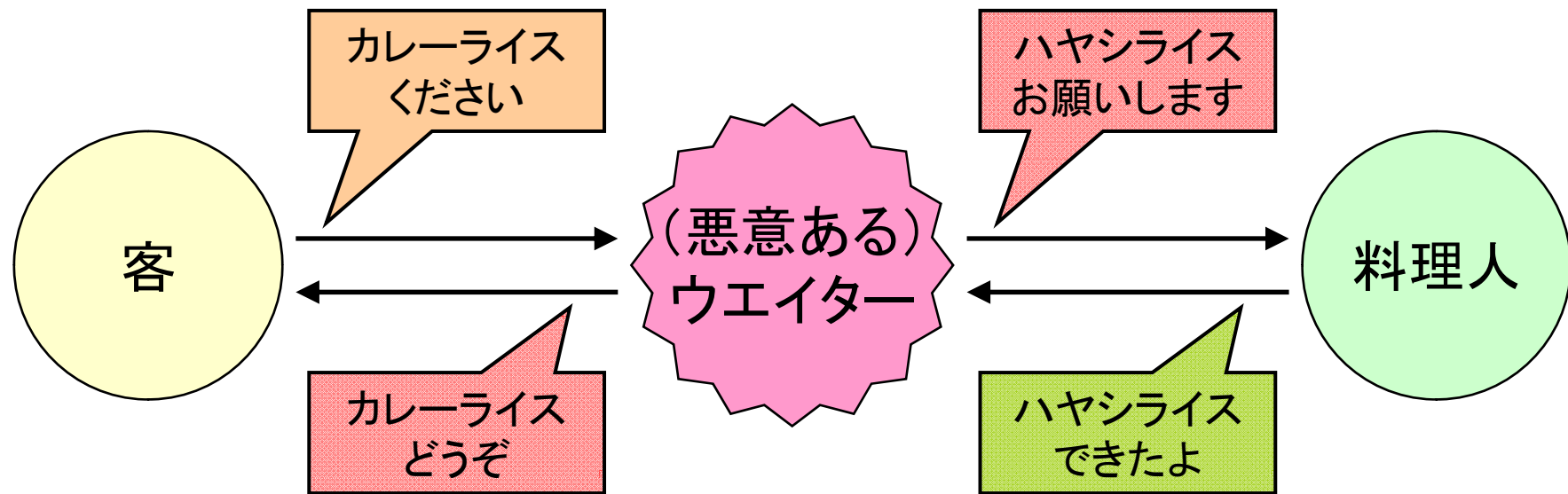
- r : セッション鍵 (実行のたびに値が変わる)
- c : AliceがBobに送りたい内容

配布案は安全か？

- 目標: Malloryが, AliceになりすましてBobと通信できるための情報(セッション鍵 r)を獲得すること

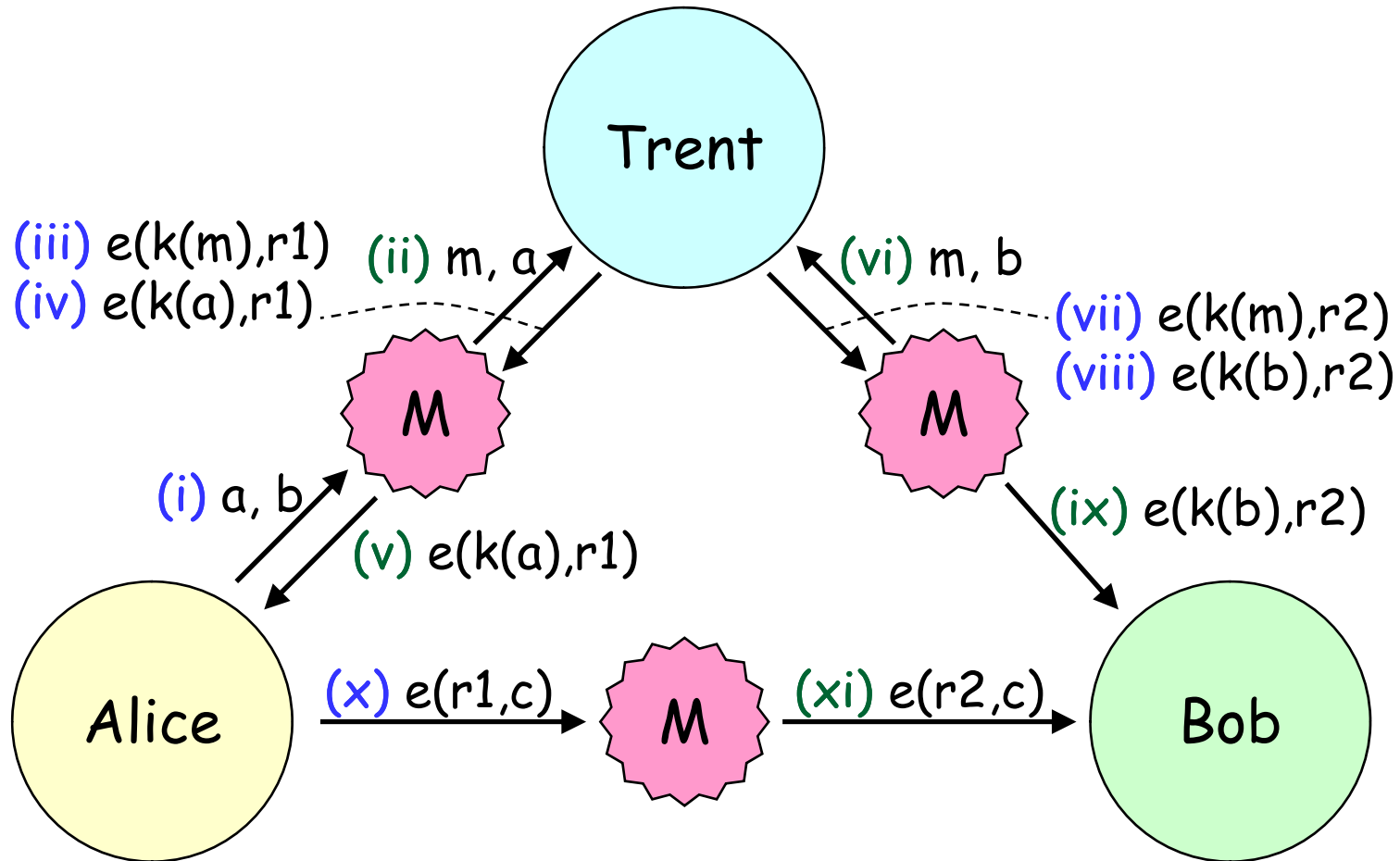


Man-in-the-middle攻撃(復習)



- Malloryがこの方法で盗聴とメッセージの改竄が行えるなら、目標を達成できてしまう.

配布案に対するMan-in-the-middle攻撃



- セッション鍵 $r1$ は, AliceとMalloryが共有する.
- セッション鍵 $r2$ は, BobとMalloryが共有する.

Feige-Fiat-Shamirの認証プロトコル

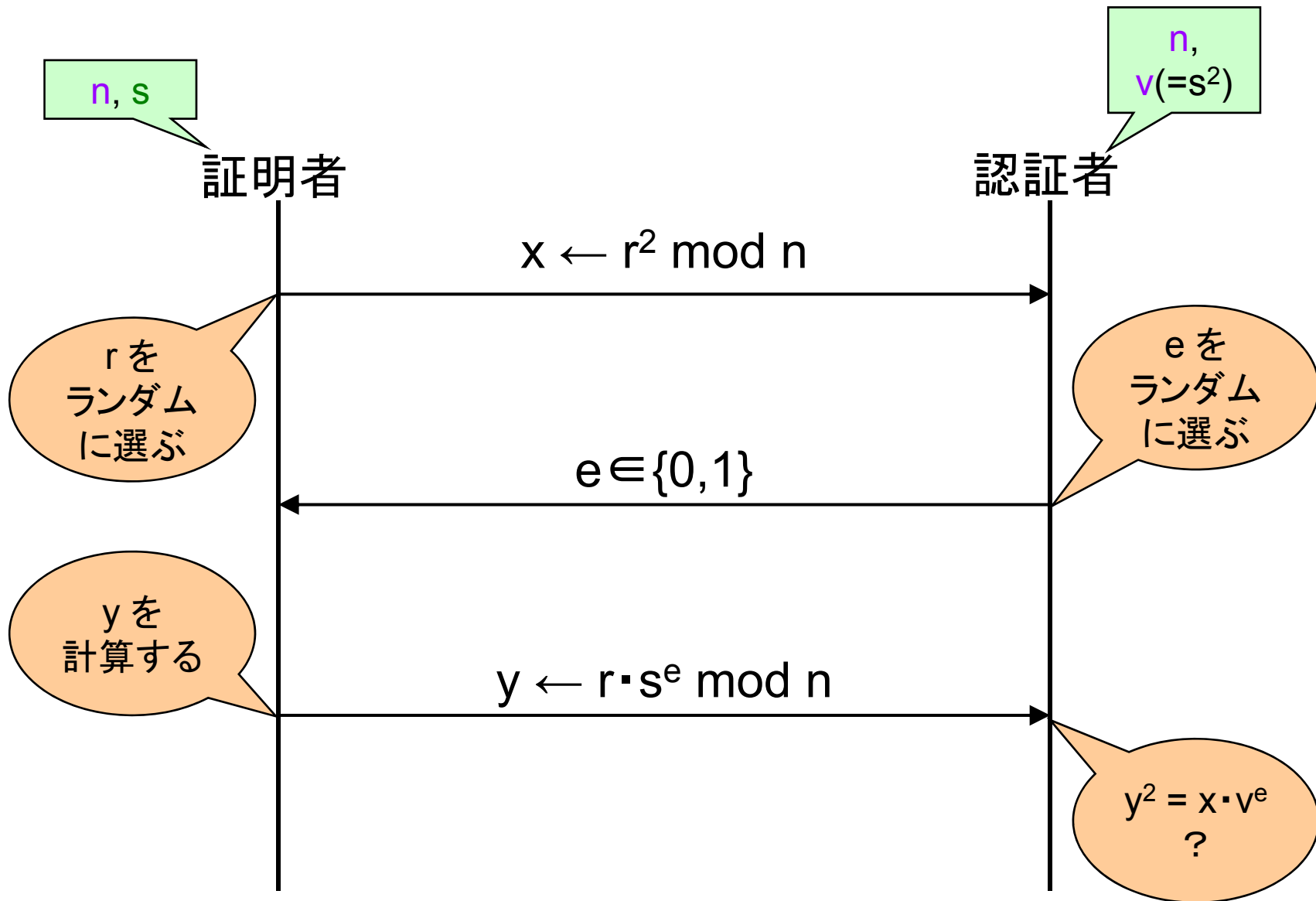
- 「FFS方式」「Fiat-Shamir方式」などとも呼ばれる
- 秘密の情報を持っていれば、プロトコルの実行により間違いなく認証者はそれを確認できる。
 - 秘密の情報は認証者にも知られない...**ゼロ知識対話証明**
(Zero-Knowledge Interactive Proof, ZKIP)
- 秘密の情報を持っていない者が、なりすましてプロトコルを実行しても、成功する確率はせいぜい1/2
 - 繰り返し実行すると、成功率は指数的に小さくなる。
- 処理速度はRSAよりも非常に速い。



FFSプロトコルの前提

- 事前に生成しておく情報
 - 素数 p, q は非公開とし, $n \leftarrow pq$ を公開
 - 証明者は整数 s ($1 \leq s < n$) を秘密情報として持つ
 - 整数 $v \leftarrow s^2 \bmod n$ も公開する
- 計算の困難さ
 - 素因数分解は困難
 - 大きな整数 m と整数 a ($1 \leq a < m$) が与えられたときに, $b^2 \bmod m = a$ を満たす b (m を法とする a の平方根) を求めるのは
 - m が素数ならば容易 (効率のよいアルゴリズムが存在する)
 - m が合成数ならば容易ではない

FFSプロトコル(図)



FFSプロトコル(記述)

- Step 1: 証明者は乱数 r を生成し, $x \leftarrow r^2 \bmod n$ を計算して, x を認証者に送る.
- Step 2: 認証者は $e \in \{0, 1\}$ をランダムに選び, 証明者に送る.
- Step 3: 証明者は $y \leftarrow r \cdot s^e \bmod n$ を計算して認証者に送る.
 - $e=0 \Rightarrow y \leftarrow r \bmod n$
 - $e=1 \Rightarrow y \leftarrow rs \bmod n$
- Step 4: 認証者は $y^2 \bmod n = x \cdot v^e \bmod n$ が成り立つか確かめ, 成り立たなければ認証しない.
 - $e=0 \Rightarrow y^2 \bmod n = r^2 \bmod n$?
 - $e=1 \Rightarrow y^2 \bmod n = r^2 s^2 \bmod n = (r^2 \bmod n)(s^2 \bmod n) \bmod n$?
 - 成り立てば, 認証者が納得いくまでStep1~4を行う.

なぜFFSプロトコルでうまくいく？

- 攻撃者の目標: 整数 s を持たないが, 証明者になりすまして認証者から認証されること
- 攻撃者があらかじめ x と y の組を生成しておけば?
 - y をランダムに生成してから, $x \leftarrow y \cdot v^{-1} \pmod n$ を作ると, $e=1$ のときは成功するが, $e=0$ のときは失敗する (n を法とする x の平方根は求められない).
- 攻撃者が過去に盗聴した情報を送れば (リプレイ攻撃)?
 - 以前の通信と e が異なっていれば失敗する.
 - すべての (x, e, y) を保存するのが現実的に難しくなるよう, p, q の大きさを決めればよい.

プロトコルのまとめ

- 基礎となる暗号技術が安全であっても、それを用いた暗号プロトコルが安全であるとは限らない.
- 暗号プロトコルに限らず、多数の人が関わるシステムを設計するときは
 - 実行環境には誰がいて、それぞれ何ができて何ができず、何をしたいかを明確にする.
 - 「うまくいく」根拠を明確にする.
 - 悪意のある者や何らかのトラブルも考慮に入れ、それでも安全な環境を維持できるようにする.