非接触型 IC カードを用いた認証プロトコル SPAIC の研究

060428262　宮崎 雄介
渡邊研究室

1. はじめに

インターネットの発展に伴い，ユーザがクライアント端末を利用して遠隔地のサーバと情報交換したいという要求が高まっている．クライアント/サーバ間通信において重要な情報を交換する場合，確実な認証と暗号化が要求される．このような要求を満たす方式として，IC カードを用いた方式が注目されており，その一技術として，SPAIC（Secure Protocol for Authentication with IC card）[1]がある．しかし，クライアント/サーバ間でのセキュリティの確認が十分に検証されていなかった．そこで，本論文では SPAIC の動作処理を見直し，さらに実装と性能評価を行った．

2. SPAIC

2.1 概要

SPAIC は非接触 IC カードを利用して，秘密情報を一切持たないクライアントに対して重要情報を配送することを可能とするオリジナルのプロトコルである．認証に必要となる初期情報はすべて IC カードに格納しているため，秘密情報が漏洩する心配がなく，特定の端末を使用する必要がないという特徴がある．

2.2 認証動作概要

見直し後の認証動作概要を図 1 に示す．第一段階はユーザ認証である．（1）ユーザが IC カードをクライアントにかざすと，IC カード公開鍵 PuI，サーバ公開鍵 PuS がクライアントに送信される．（2）ユーザはパスワードを入力し，これをクライアントは IC カード公開鍵で暗号化する．更に Diffie-Hellman 鍵交換の交換値（DH1）を生成し，これらの情報を IC カードへ送信する．（3）IC カードは，IC カード秘密鍵を用いて復号し，パスワードの照合を行うことで認証する．

第二段階は，IC カード認証である．（3）IC カードは IC カードの秘密鍵を用いて署名情報 S_I（DH1）を作成し，クライアントを経由して，サーバへ送信する．（4）サーバは，IC カード公開鍵を用いてディジタル署名の検証することで認証する．IC カードはユーザを認証済みなので，間接的にユーザが使用しているクライアントを認証したことになる．

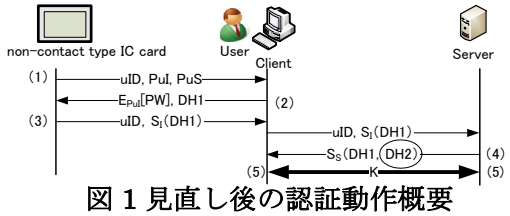


図 1 見直し後の認証動作概要

第三段階は，サーバ認証である．（4）サーバは DH 交換値（DH2）を生成し，サーバの秘密鍵を用いてディジタル署名 S_S(DH1, DH2)を作成し，クライアントへ送信する．（5）クライアントは，サーバ公開鍵を用いてディジタル署名の検証を行い認証する．

以上の三段階の認証により，クライアント/サーバ間の認証が完了する．上記手順の中で DH1，DH2 の共有が行われているため，クライアント，サーバは共通暗号鍵 K を生成できる．以降のクライアント/サーバ間の通信はこの暗号鍵 K を用いて行う．

これまで，攻撃者がサーバとの間で共通暗号鍵の取得ができてしまう問題が有ったため，図中○印の DH2 を追加し，プロトコルを完成させた．

3. 実装

現段階で公開鍵演算処理が可能な非接触型 IC カードの入手は困難である．そこで，代替として USB トークンによる試作を行うこととした．USB トークンは，IC カードと同様にスマートカードリーダとして Windows システム（デバイスマネージャ）上で認識される．USB トークンにはプロセッサとメモリが搭載されており，内部で RSA 演算処理が行える．

現在の実装状況は，1 対 1 のメッセージ交換処理が可能であり，各種エラー処理と複数のクライアントからの接続要求処理に伴う処理は未実装となっている。

4. 評価

SPAIC を実現する上でボトルネックとなるのは，公開鍵暗号の処理時間である．そこで，全体の処理時間の多くを占める公開鍵暗号に要する時間を求め，全体の処理時間を推測した．

各暗号化処理は，RSA 暗号の 1024bit とし，1000 回試行した平均値を求めた．パソコン上での暗号化・署名の処理時間は 166.9μs，復号・署名の確認は 160.3μs であり，USB トークンは 293.8ms と 293.9ms であった．よって，全体に掛かる処理はおよそ 1 秒程度となる．立ち上げ時に掛かる処理としては，許容範囲と考える．

5. むすび

本論文では，SPAIC のシーケンスを見直し，セキュリティの向上を行った．さらに，USB トークンを用いた実装と評価を行い，立ち上げ時の認証において，十分に許容できることを確認した．

参考文献

[1] 東長俊, 鈴木秀和, 渡邊晃：非接触型 IC カードを用いた認証方式 SPAIC の提案，DICOMO2007, pp.1332-1337 (2007).

非接触型ICカードを用いた認証プロトコルSPAICの研究

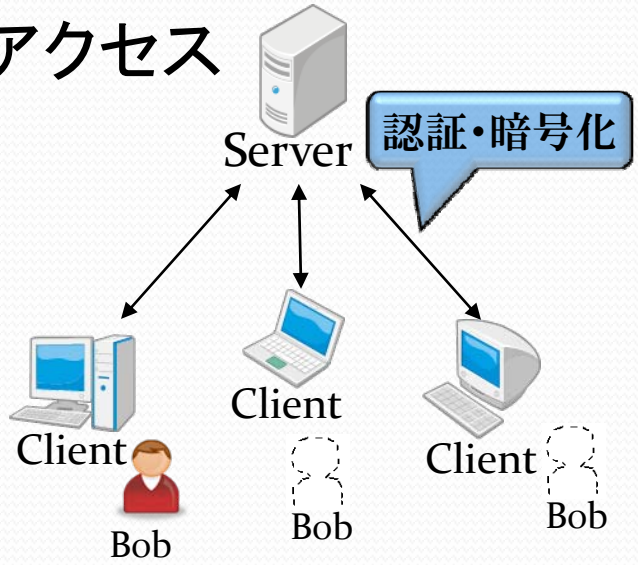
渡邊研究室

060428262

宮崎 雄介

研究背景

- クライアント/サーバ間通信の安全確保
  - 重要情報の漏洩を防ぐ
- 異なるクライアントからのサーバへのアクセス
  - 例：自宅や学校、あるいは会社など



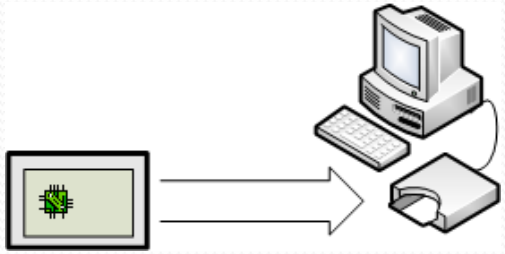
確実な認証と暗号化が必要

ICカードを利用した認証方式に注目

- カード内で認証や暗号化などの処理が可能
- 外部からの不正読み取りを防ぐことができる（耐タンパ性）
- 一人一人が持って移動できる（利便性）

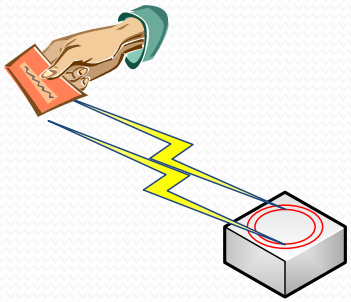
ICカードの分類

- 接触型ICカード
  - ICカード/クライアント間は一体
    - 一般的に、暗号化を行わない



ICカードをリーダライタに挿入する

- 非接触型ICカード
  - ICカード/クライアント間は無線通信
    - 暗号化が必要



ICカードをリーダライタにかざす

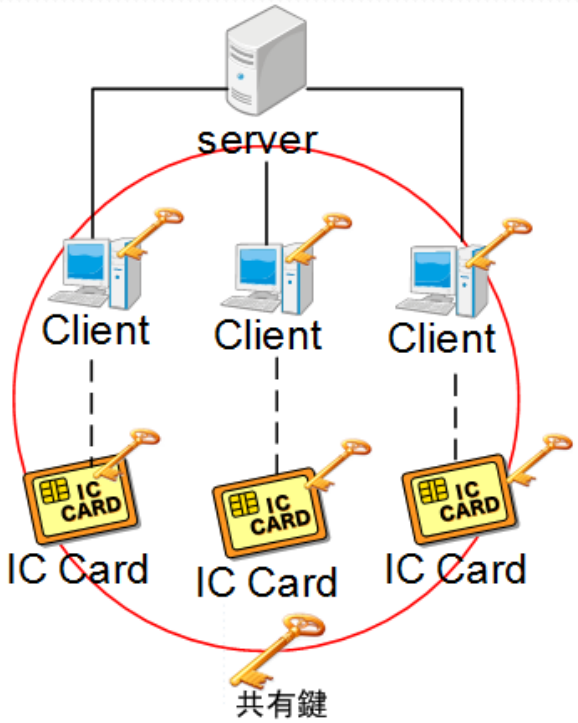
既存技術と課題

- 既存技術：事前共有鍵方式→JICSAP*で定義
  - 事前に共有鍵を全てのIC CardとClientで共有する
  - 共有鍵を用いて暗号化キーを生成する
  - ICカード/クライアント間を暗号化

- 課題
  - クライアントから共有鍵が漏洩
    - 影響が全体に及ぶ
  - 共有鍵の更新が定期的に必要となる
    - 鍵の管理が煩雑

解決

SPAIC



*JICSAP:日本ICカードシステム利用促進協議会

SPAICについて

- SPAIC:Secure Protocol for Authentication with IC Card

目的

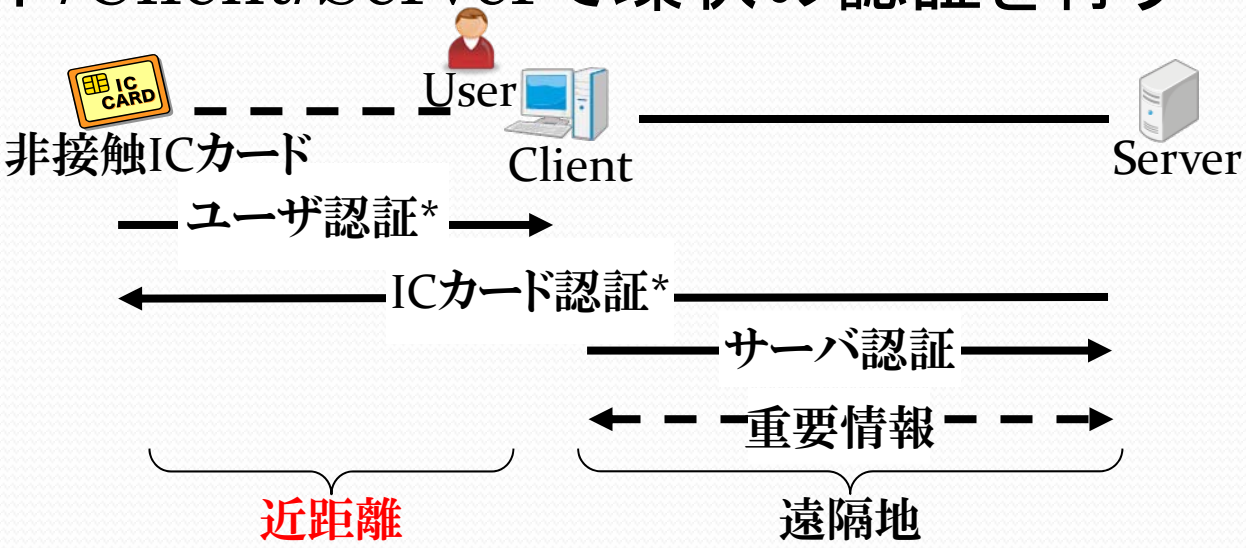
- 非接触ICカードを利用し、ServerからClientへ重要情報を安全に配送するための通信路を確立する

特徴

- クライアントに初期情報を一切所持しない
  - 情報漏えい防止
- ICカード/クライアント間の認証に
  - ICカードの公開鍵を利用

SPAICの認証システム概要

- 前提条件
  - ☆ICカード/クライアント間は、近距離であるため中間者攻撃が出来ない
    - ☆クライアントに初期情報が無くても可能にする
- ICカード/Client/Serverで環状の認証を行う



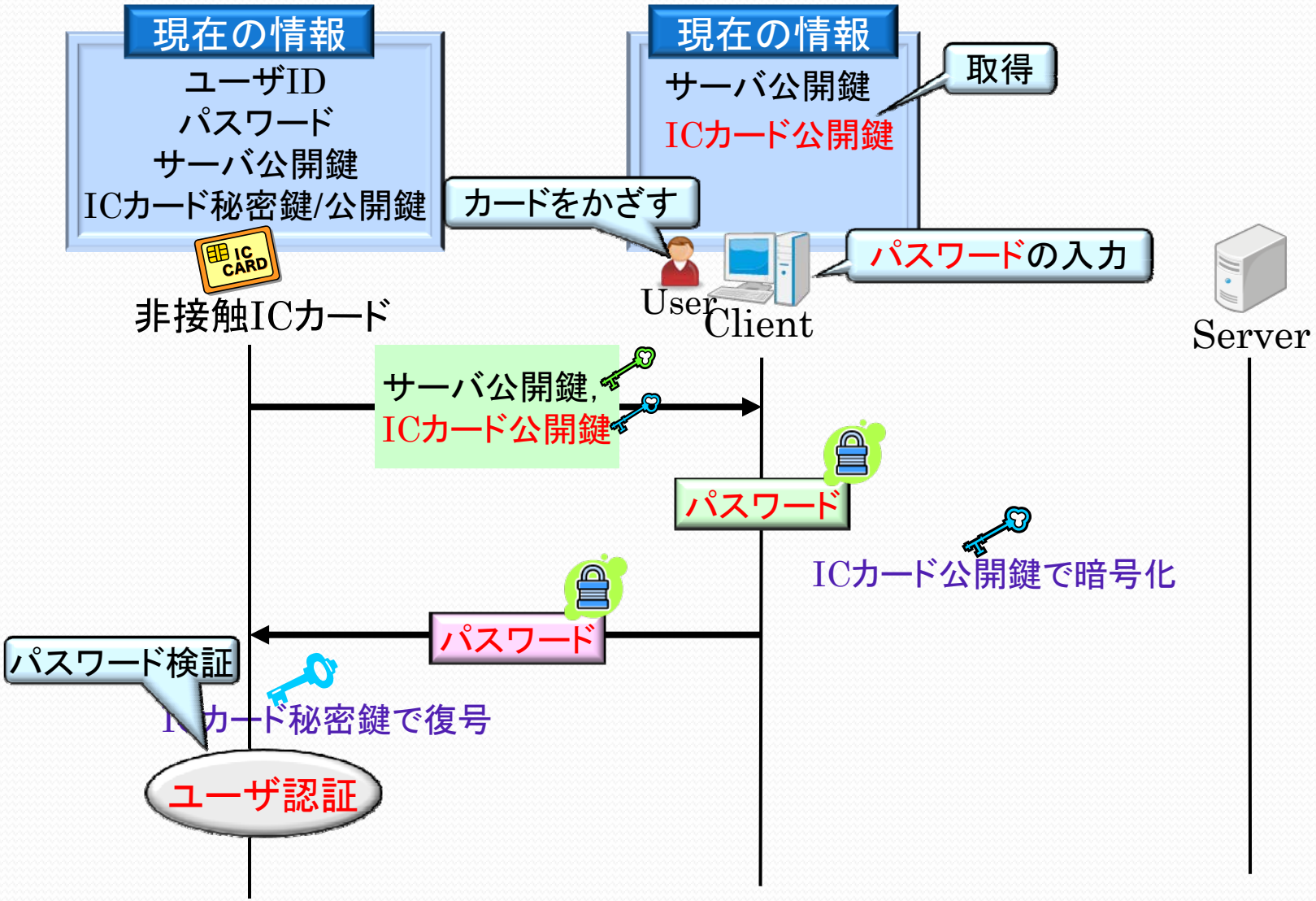
*クライアントは間接的に認証

各端末の初期情報

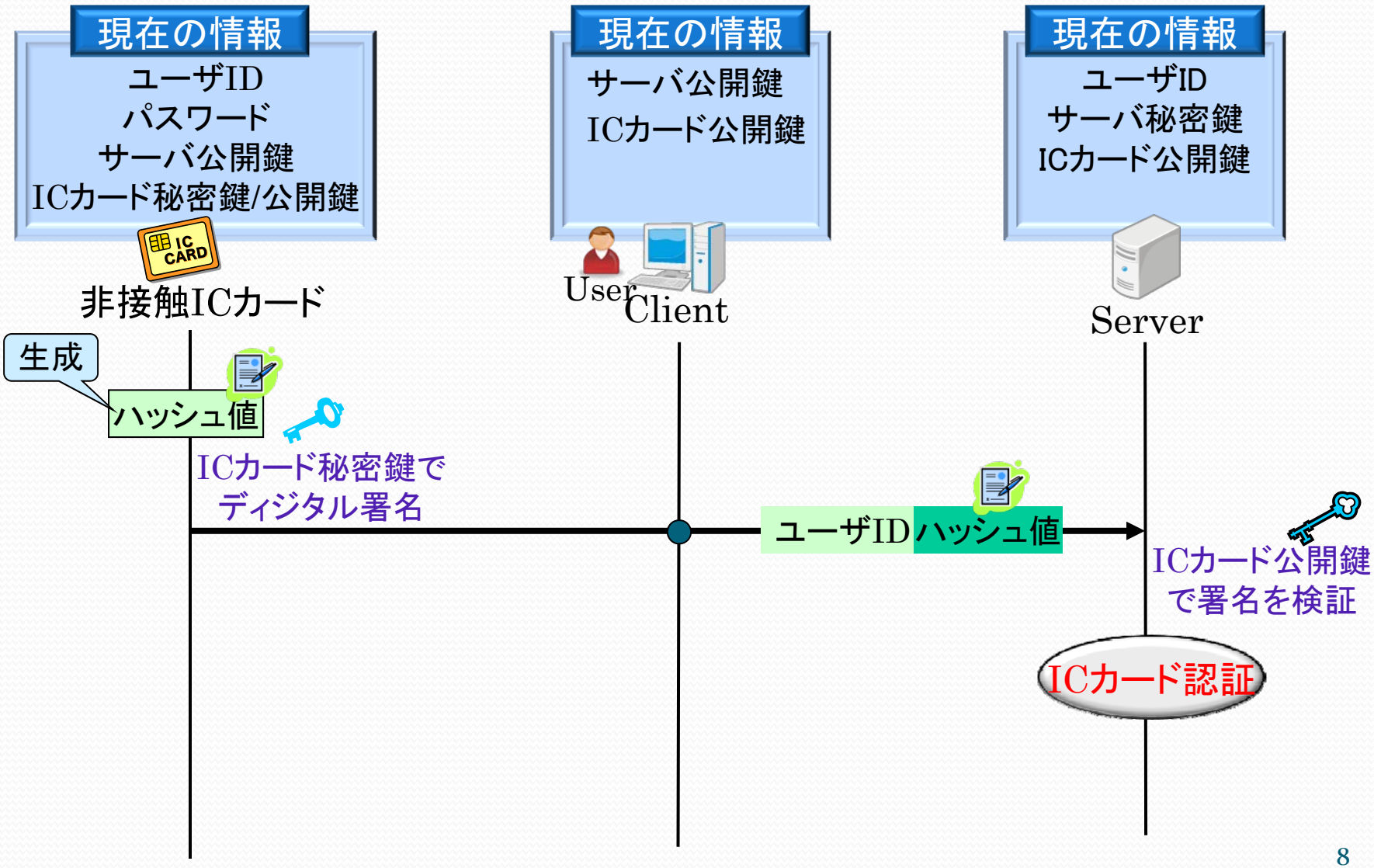
	事前共有鍵方式（既存）	SPAIC(提案)
 ICカード	ユーザID パスワード サーバ公開鍵 ICカード秘密鍵 事前共有鍵	ユーザID パスワード サーバ公開鍵 ICカード秘密鍵 ICカード公開鍵
 クライアント	事前共有鍵	なし
 サーバ	ユーザID サーバ秘密鍵 ICカード公開鍵	ユーザID サーバ秘密鍵 ICカード公開鍵

- 初期情報は事前にオフラインで設定する

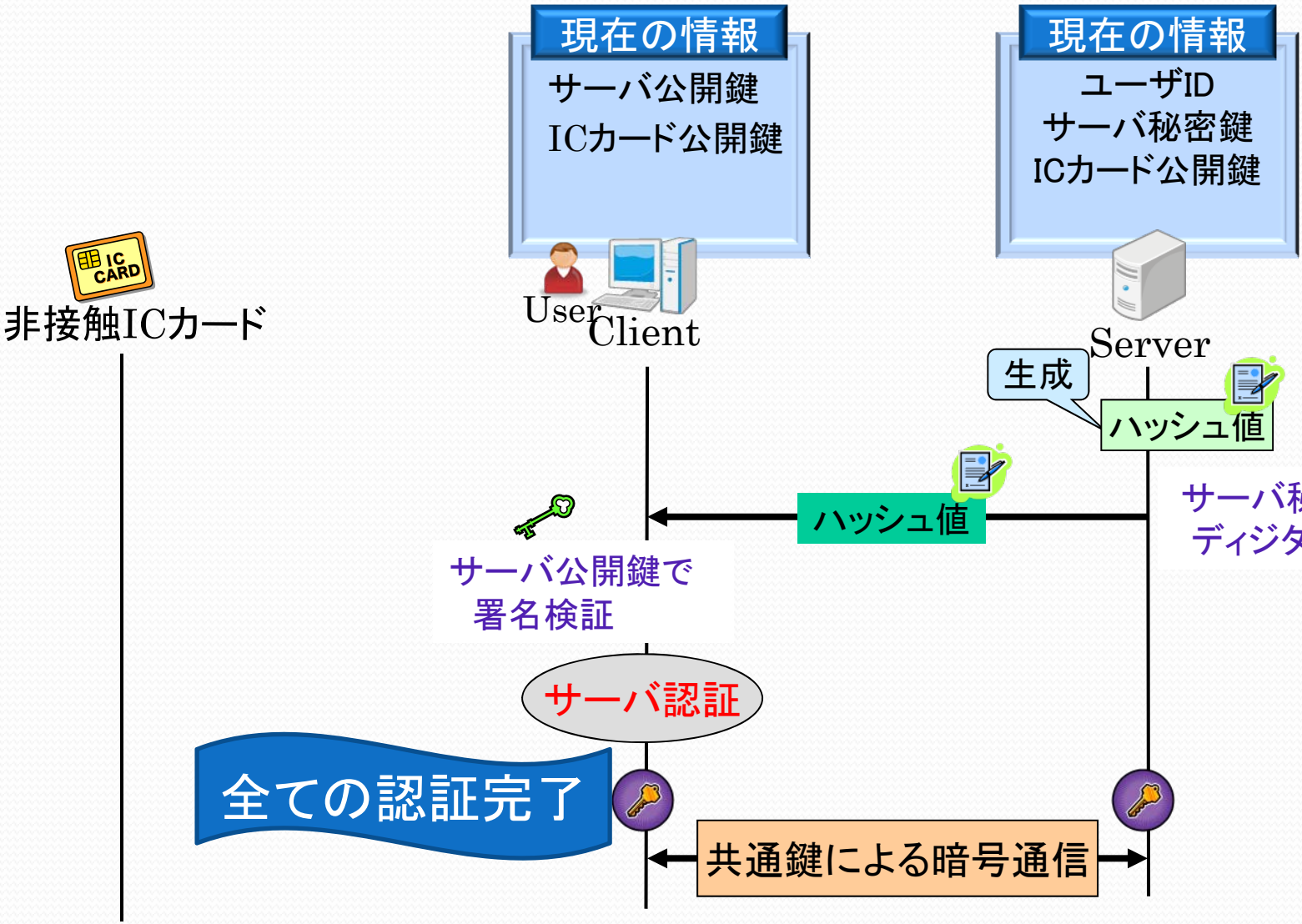
SPAICの動作1＜ユーザ認証＞



SPAICの動作2＜ICカード認証＞



SPAICの動作3＜サーバ認証＞



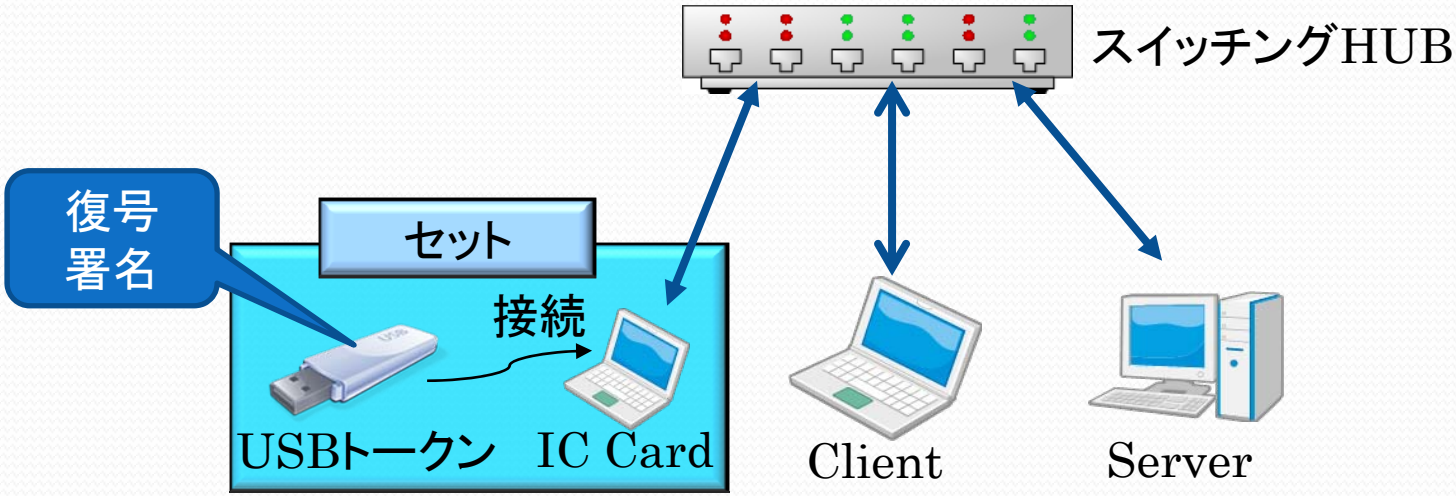
実装

- 実験の目的
  - 動作検証
  - 性能評価
- USBトークン
  - 公開鍵演算が可能な非接触型ICカードの入手が困難
  - 演算処理が可能なUSB

暗号化・復号　電子署名の作成　乱数生成　鍵の格納

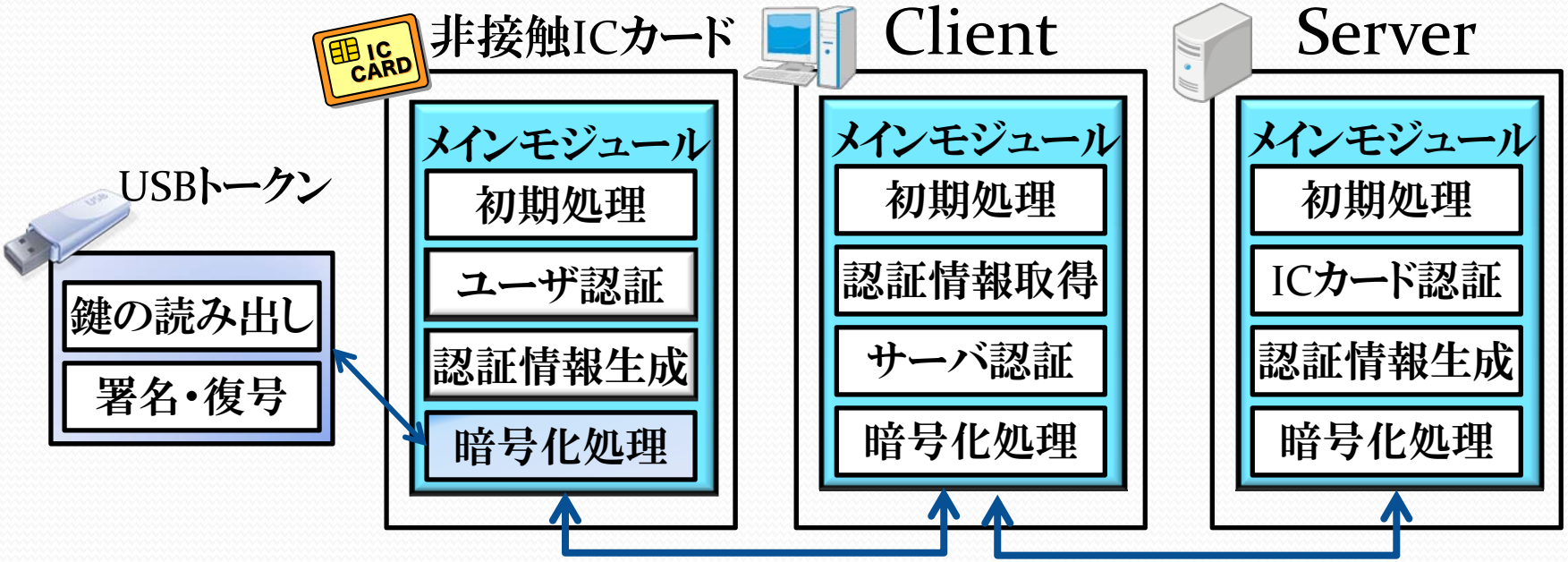
ICカードの代用として使用可能

実験環境



装置名	スペック		
	CPU	Memory	OS
ICカード(PC1)	Pentium M(1．7GHz)	504MB	XP professional SP3
Client(PC2)	Core 2 Duo U7600(1．2GHz)	2GB	7　Ultimate
Server(PC3)	Core 2 Duo E 6600(2．4GHz)	4GB	Vista Business SP2
USB トークン	Sony　FIU-810-N03 ARM7		

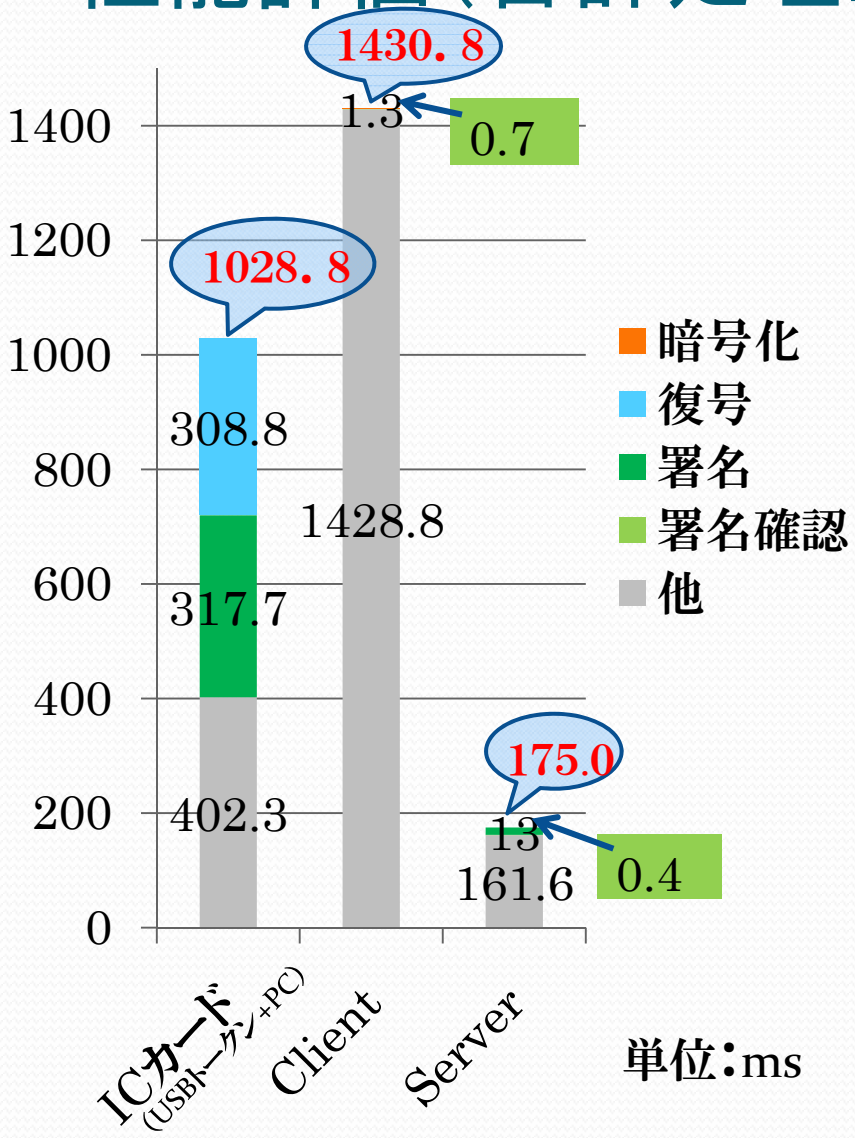
モジュール構成



- ICカードの暗号化処理はUSBトークンが行う
  - 他のICカードが行うべき処理はPC上で行われる
- PC上の暗号化処理にはOPENSSL*のソースを利用
- 暗号化にはRSAの1024bitを用いている

*暗号化関数と様々なユーティリティ関数を実装しているオープンソース

性能評価（合計処理時間と公開鍵演算時間）



- 計測には、Wireshark*とQueryPerformanceCounter関数を使用
- ICカードの合計処理時間は1028．8ms

認証処理完了時間：約2．6秒

認証動作（SPAIC）は、
立ち上げ時に掛かる処理
なので、許容範囲と考える

*パケット・アナライザ・ソフト

まとめ

- SPAICは
  - 非接触ICカードを利用し、クライアント/サーバ間の安全な通信路を確立する
  - クライアントに初期情報が必要ない
- USBトークンによる試作システムの実装・評価を行った

今後

- 実装の改善
  - 無駄な処理の解決
  - プログラムの効率化

Fin…

付録

暗号化技術

- 公開鍵暗号
  - 暗号化と復号に異なる鍵を使用する暗号方式
  - 一方の鍵で暗号化したデータは、もう片方の鍵でしか復号できない
  - デジタル署名による認証も可能
  - 演算に時間が掛かる
- Diffie-Hellman鍵交換
  - 乱数を通信路上で交換して、共通鍵を生成
  - 第三者が乱数を盗聴しても、鍵の取得が不可能

USBトークンについて

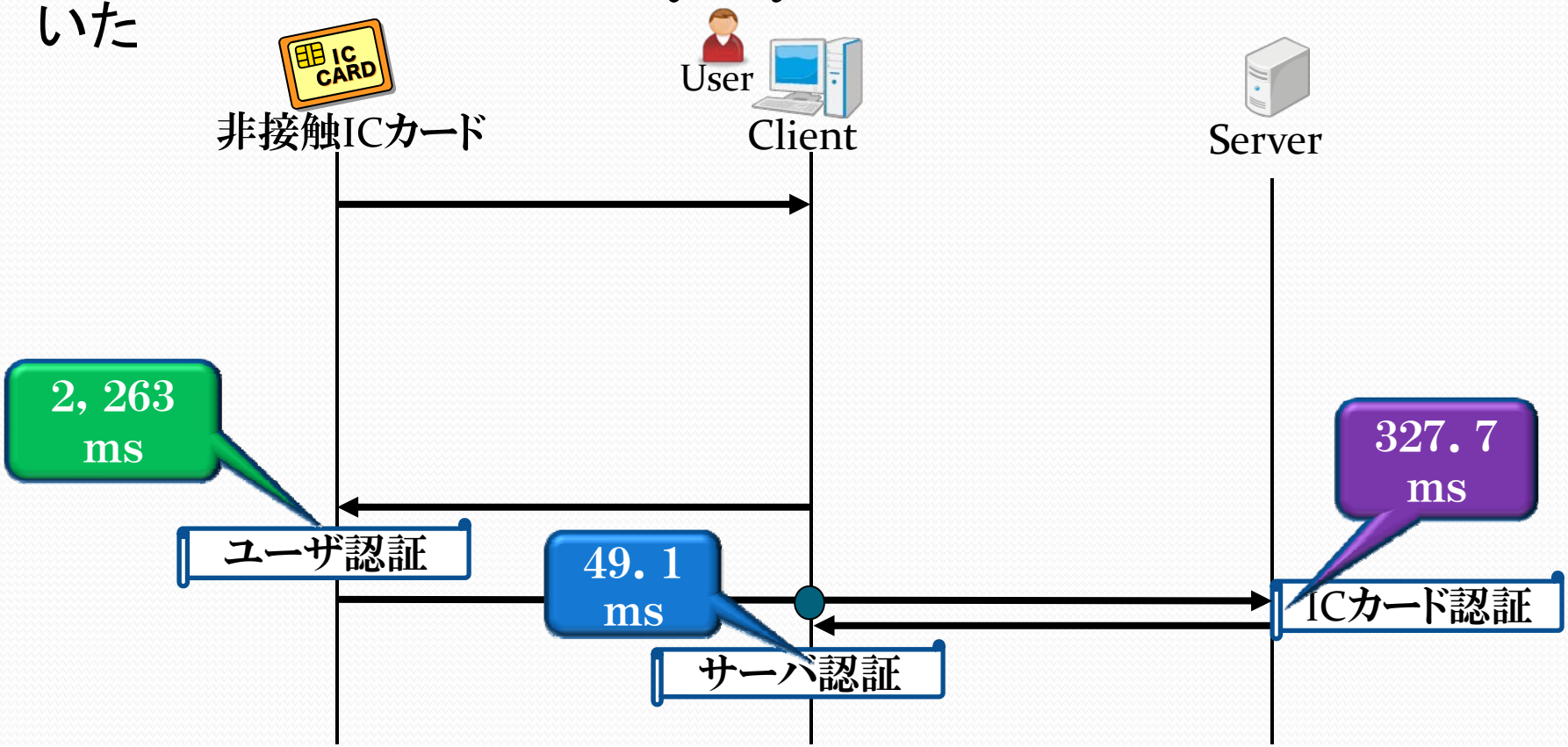
- PUPPY
- 暗号 LSI
  - CPU (ARM7), Mask ROM, SRAM, EEPROM, USB コントローラ, RSA 暗号処理エンジンで構成される
- ARM7（18～56MHz）
  - 全世界で最も使用されている32bit CPUアーキテクチャのひとつ
  - 初期の3G携帯に採用

ICカード関連企業の動向

- JICSAP（1993年設立）
  - メーカー・システムイングレーター・利用する企業など日本の主要企業の多くが参加して標準化に取り組んでいる団体
- マルトス推進協議会（2000年設立）
  - ICカード用OSがマルトス。大日本印刷や銀行、クレジット会社が集まって設立された。日本におけるマルトスの利用環境の改善を行い、マルトスの普及促進を目指している団体。

実験結果

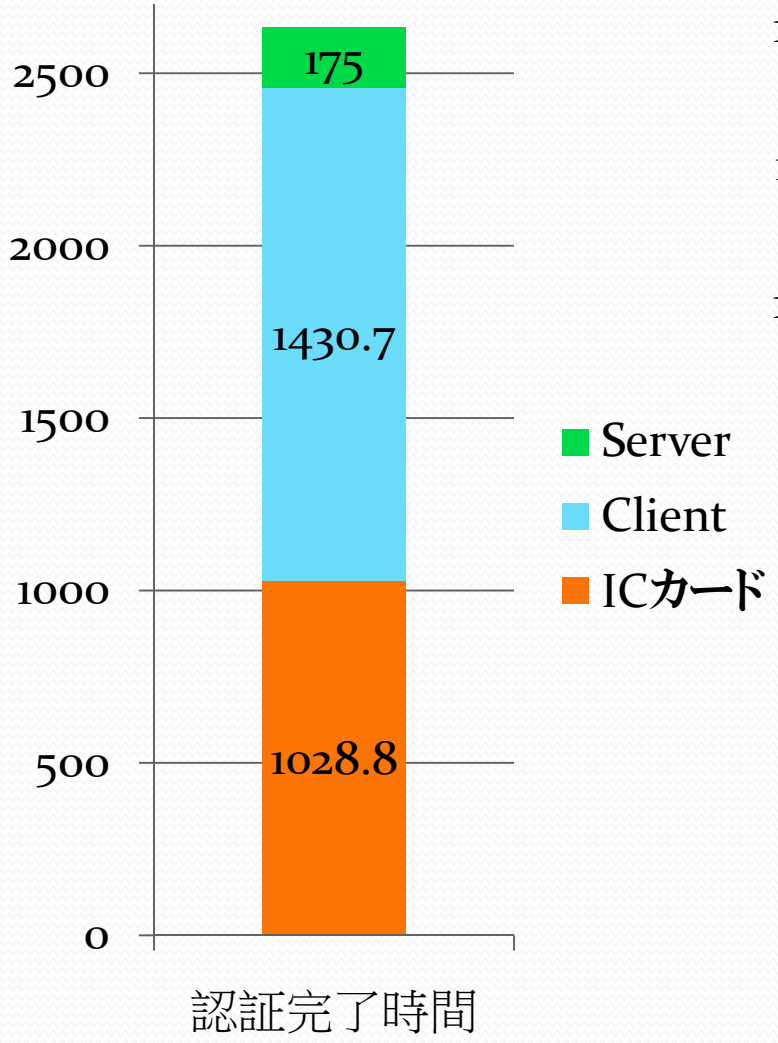
- 計測には、Wireshark*とQueryPerformanceCounter関数を用いた

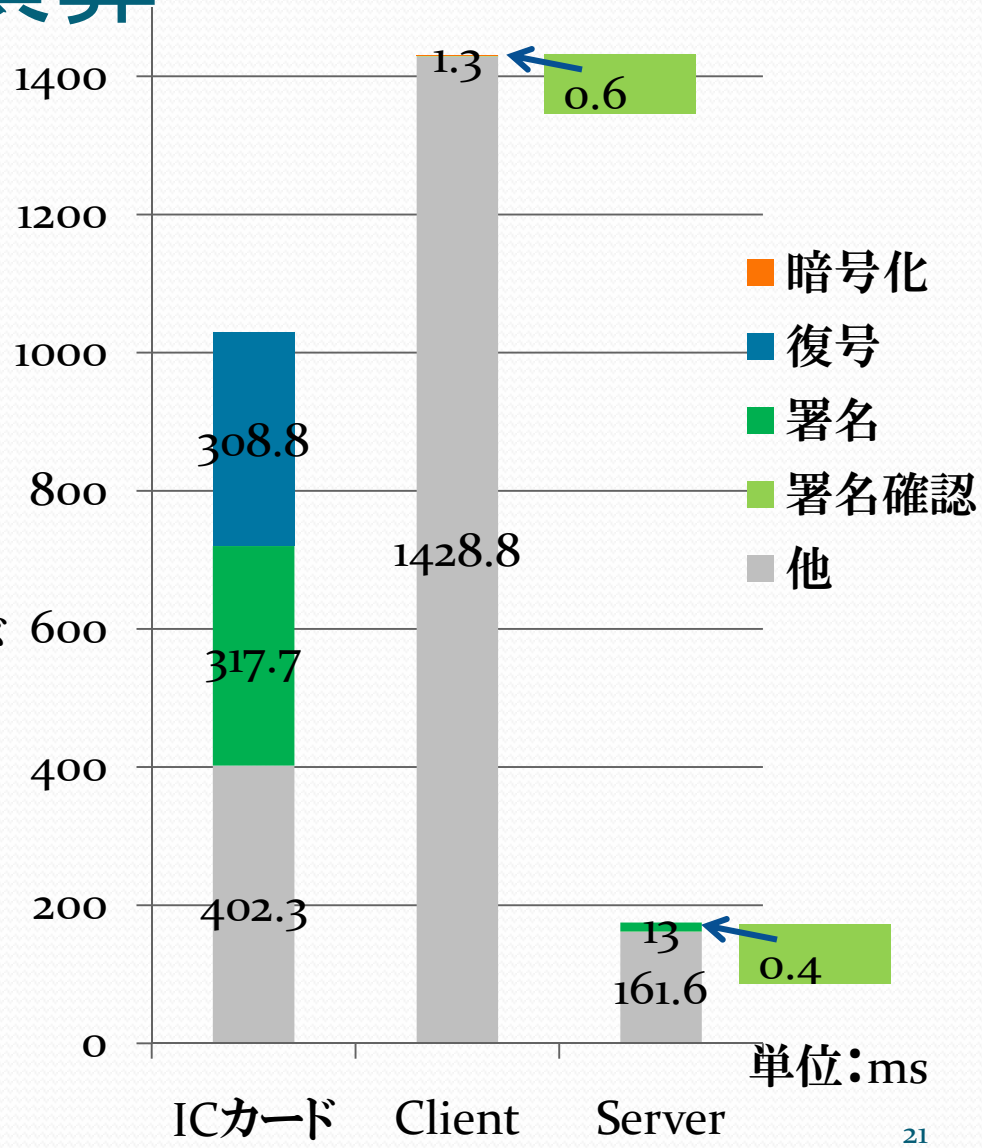


認証処理完了時間⇒2. 6秒

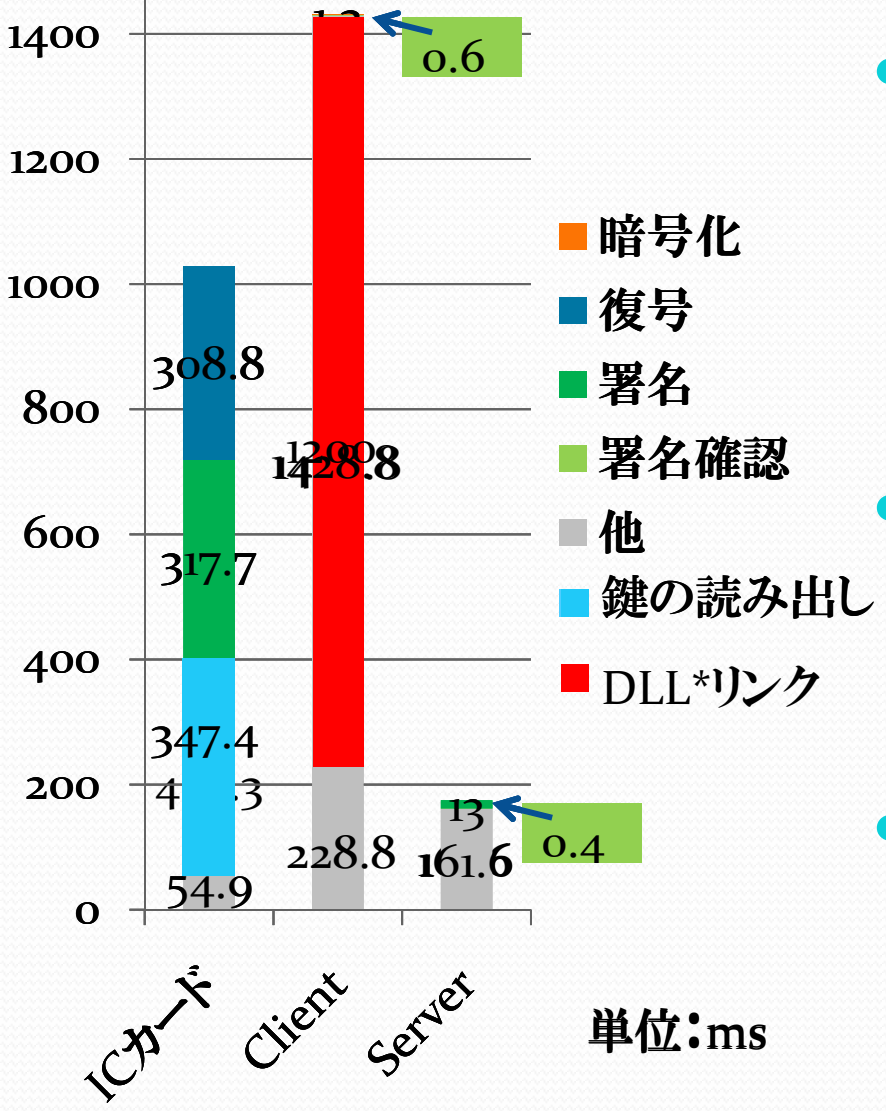
＊パケット・アナライザ・ソフト

各装置の公開鍵演算


2500
2000
1500
1000
500
0
175
1430.7
1028.8
Server
Client
ICカード
認証完了時間

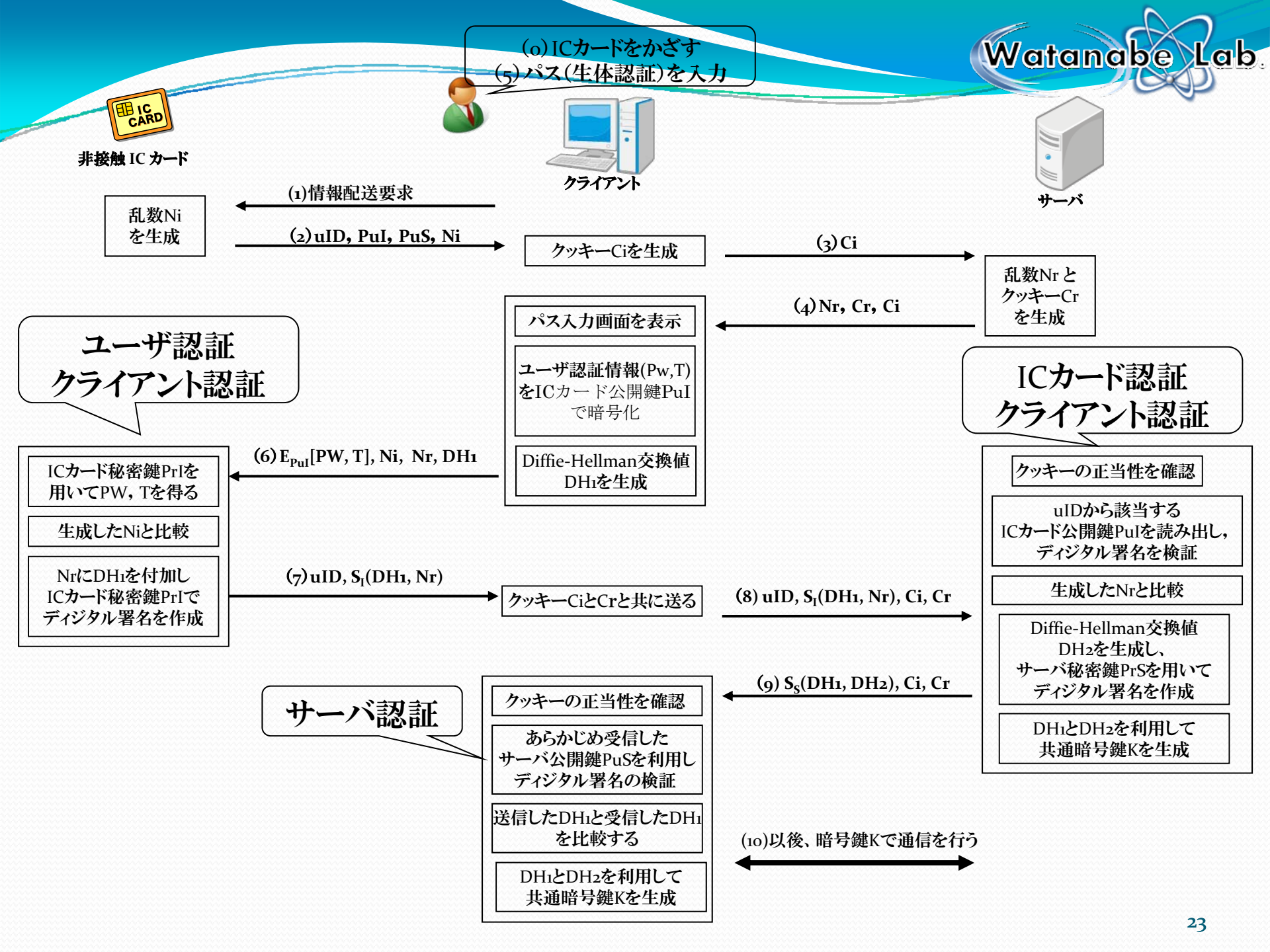

1400
1200
1000
800
600
400
200
0
308.8
317.7
402.3
1.3
1428.8
0.6
13
161.6
0.4
暗号化
復号
署名
署名確認
他
単位:ms
ICカード
Client
Server

考察



- ICカードの認証処理に2秒以上
- 原因
  - ① USBトークンから鍵を読み出すために347．4ms
  - ② OPENSSLの関数を使用する初回にDLLの読み込みが1．2秒程度
- ②を改善することで認証完了まで1．4秒程度となる
  - その内、USBトークンによる処理は973．9ms
- 認証動作（SPAIC）は、立ち上げ時に掛かる処理なので、許容範囲と考える

*Dynamic Link Library

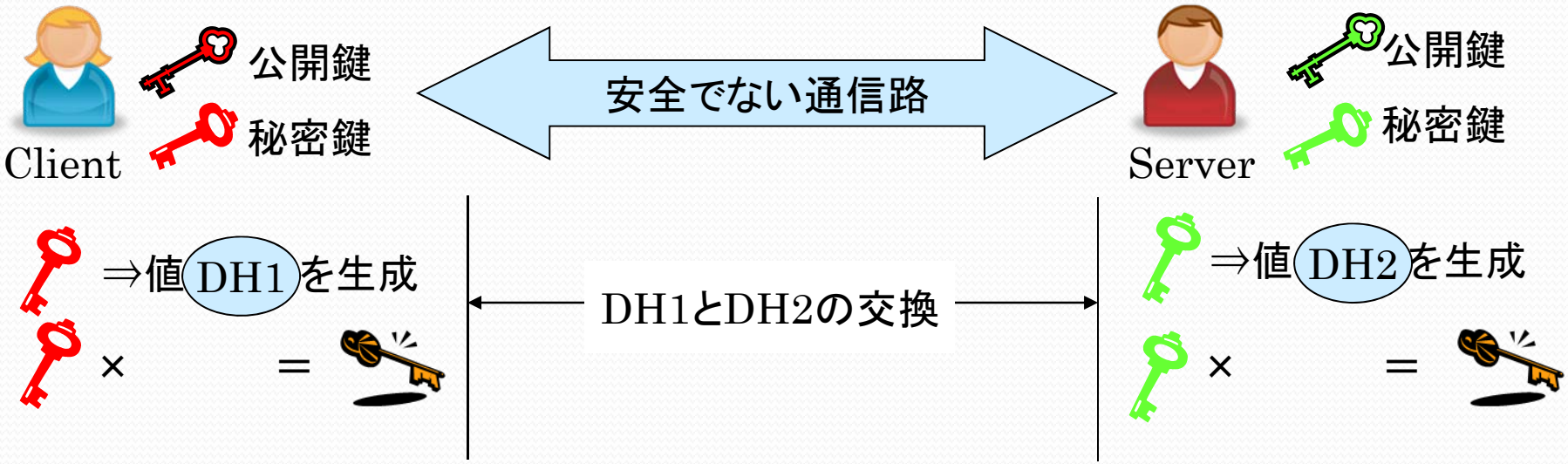

(0) ICカードをかざす
(5) パス(生体認証)を入力
非接触 IC カード
クライアント
サーバ
(1)情報配送要求
乱数Niを生成
(2) uID, PuI, PuS, Ni
クッキーCiを生成
(3) Ci
乱数Nr とクッキーCrを生成
(4) Nr, Cr, Ci
パス入力画面を表示
ユーザ認証情報(Pw,T)をICカード公開鍵PuIで暗号化
Diffie-Hellman交換値DH1を生成
ユーザ認証
クライアント認証
ICカード秘密鍵PrIを用いてPW, Tを得る
生成したNiと比較
NrにDH1を付加しICカード秘密鍵PrIでディジタル署名を作成
(6) E_PuI[PW, T], Ni, Nr, DH1
(7) uID, S_I(DH1, Nr)
クッキーCiとCrと共に送る
(8) uID, S_I(DH1, Nr), Ci, Cr
ICカード認証
クライアント認証
クッキーの正当性を確認
uIDから該当するICカード公開鍵PuIを読み出し, ディジタル署名を検証
生成したNrと比較
Diffie-Hellman交換値DH2を生成し、サーバ秘密鍵PrSを用いてディジタル署名を作成
DH1とDH2を利用して共通暗号鍵Kを生成
(9) S_S(DH1, DH2), Ci, Cr
サーバ認証
クッキーの正当性を確認
あらかじめ受信したサーバ公開鍵PuSを利用しディジタル署名の検証
送信したDH1と受信したDH1を比較する
DH1とDH2を利用して共通暗号鍵Kを生成
(10)以後、暗号鍵Kで通信を行う

SPAICのセキュリティ

- 安全な通信路の確立には様々な攻撃手法に対処する必要がある
  - 盗聴　⇒暗号化
  - Dos攻撃　⇒Cookie
  - リプレイ攻撃　⇒乱数
  - 偽造・改竄　⇒ディジタル署名

Diffie-Hellman鍵共有とは

- 事前に秘密の共有が必要なし、盗聴される可能性のある通信路を使っても、暗号鍵の共有を可能にするプロトコル

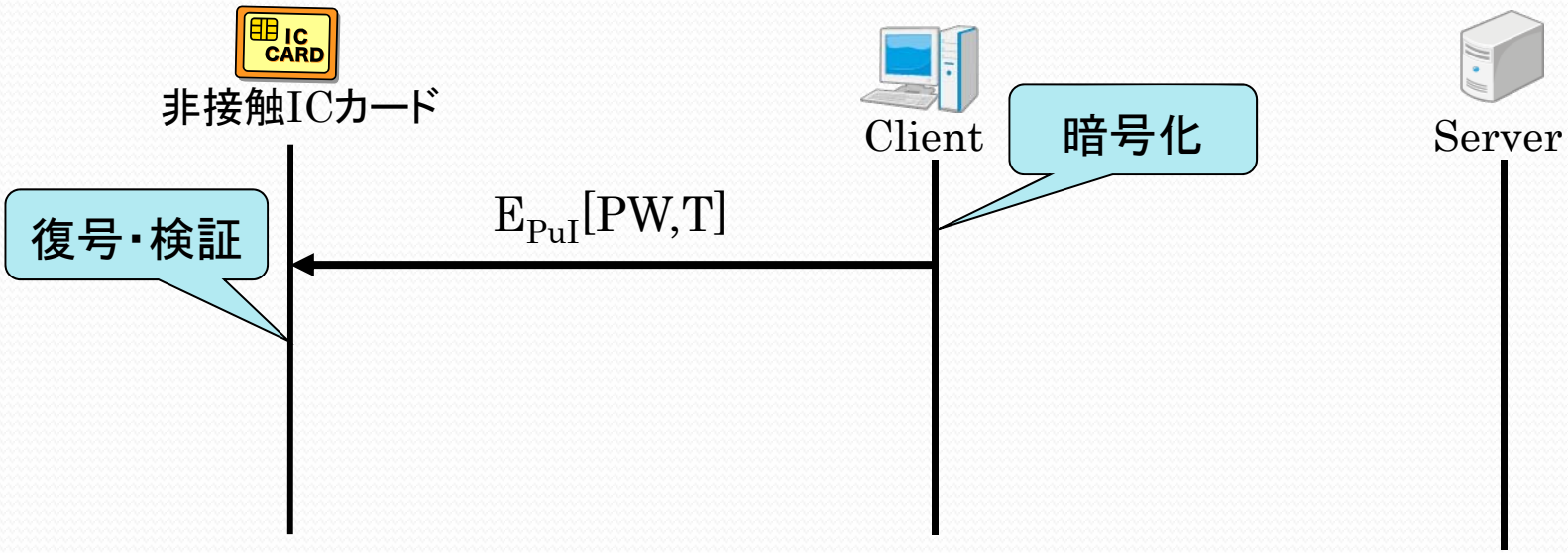


「自身が生成した値」と「その値を使用して作成された相手の値」が必要

セキュリティ1＜ 盗聴 ＞

暗号化で対応

- 盗聴
  - 通信内容を盗み見ること



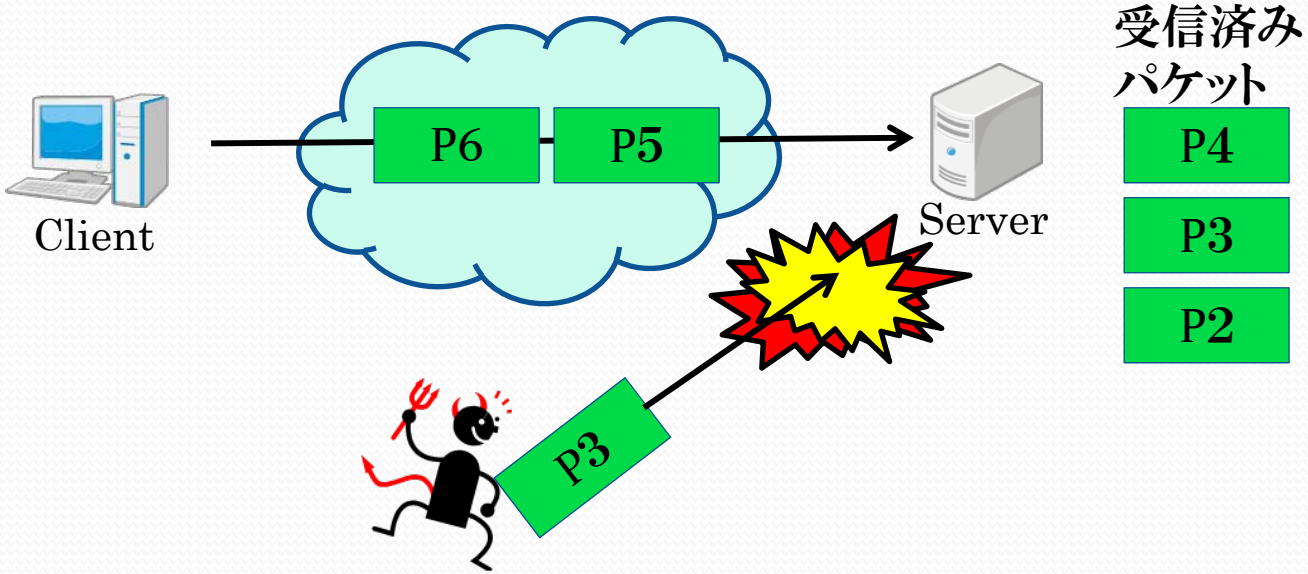
盗聴されては困るPW,Tを暗号化する＝盗聴不可

E_{PuI}[X]：XをICカードの公開鍵で暗号化

セキュリティ2＜リプレイ攻撃＞

乱数で対応

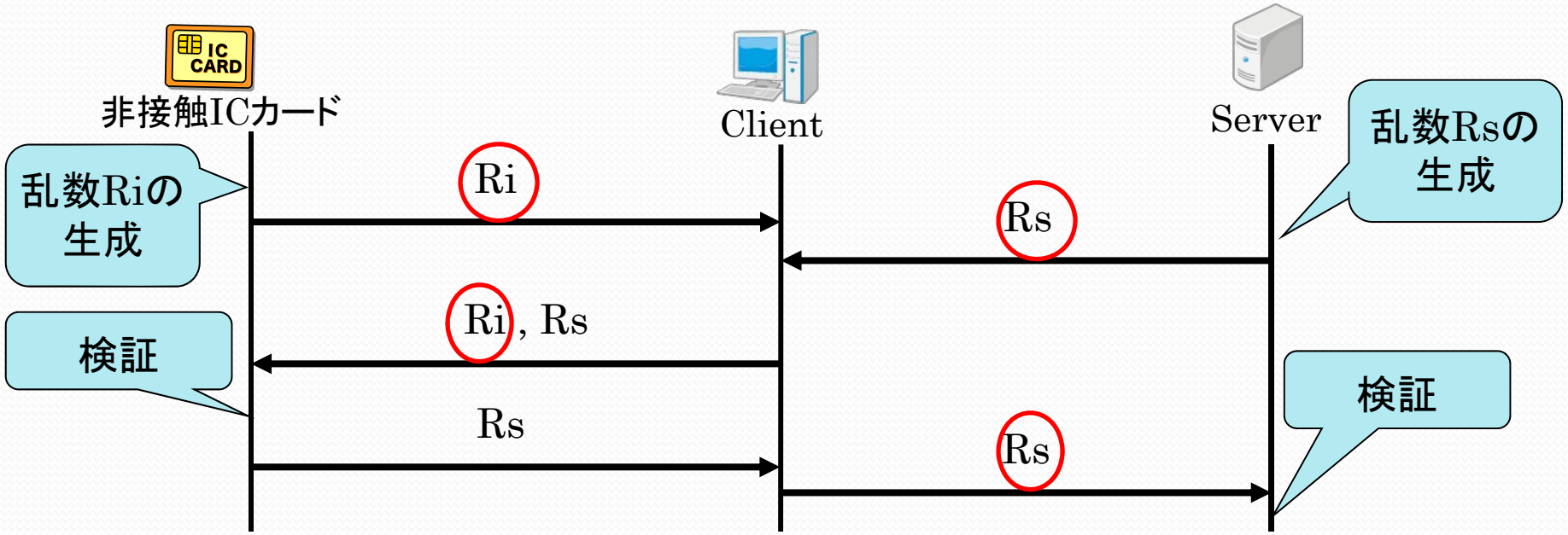
- ✓ 乱数は通信毎に生成される
- ✓ 既に受信済みパケットは拒否する
- ✓ 通信終了後サーバはパケットを破棄
- ✓ 以後、古いパットはCookienoタイムスタンプにより拒否



セキュリティ2＜リプレイ攻撃＞

- リプレイ攻撃
  - 正規の通信をコピーし、コピーしたデータを再利用することで、なりすます攻撃

乱数で対応

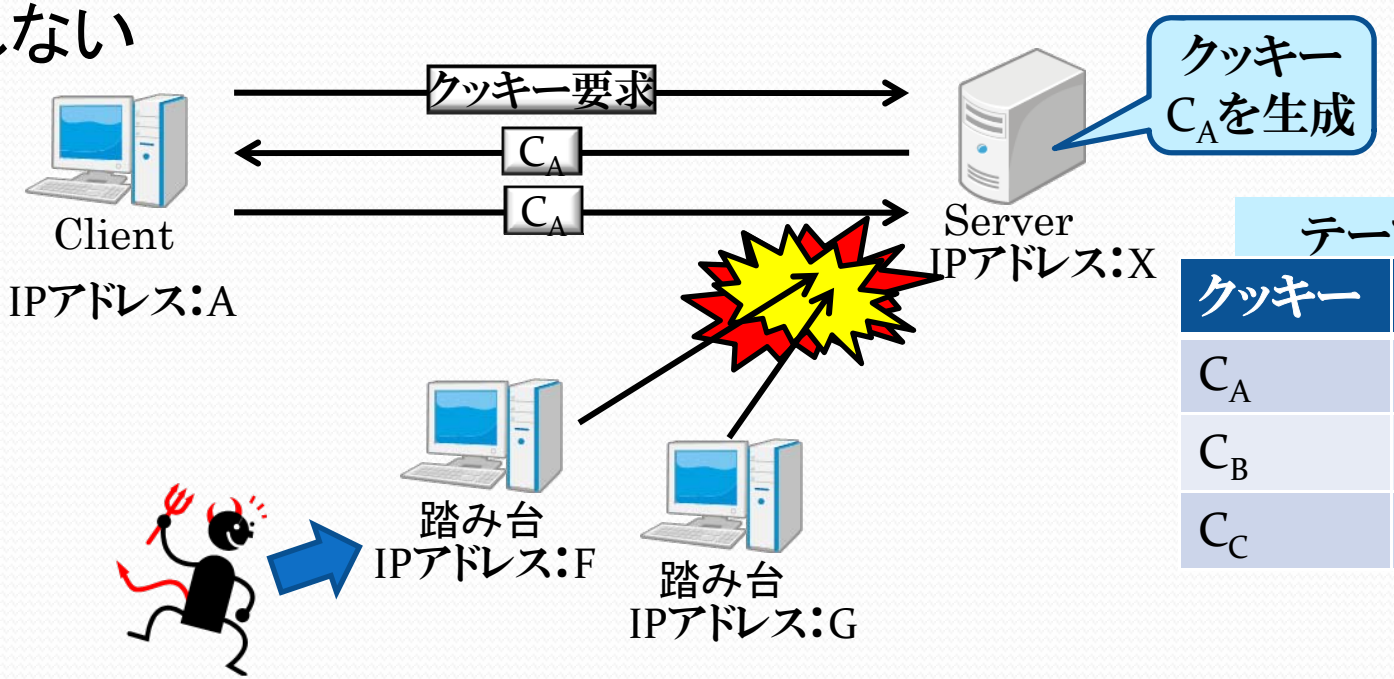


乱数は通信毎に異なる値＝リプレイ（再利用）不可

セキュリティ3＜ Dos攻撃 ＞

Cookieで対応

- Cookie（作成日時を含む）
  - 送信元IPアドレスと送信先IPアドレス、乱数を基に作成
  - 通信に先立ってCookie交換を行う（通信毎に異なる）
- IPアドレスを偽造してくるDos攻撃はテーブル表に該当しない



テーブル表

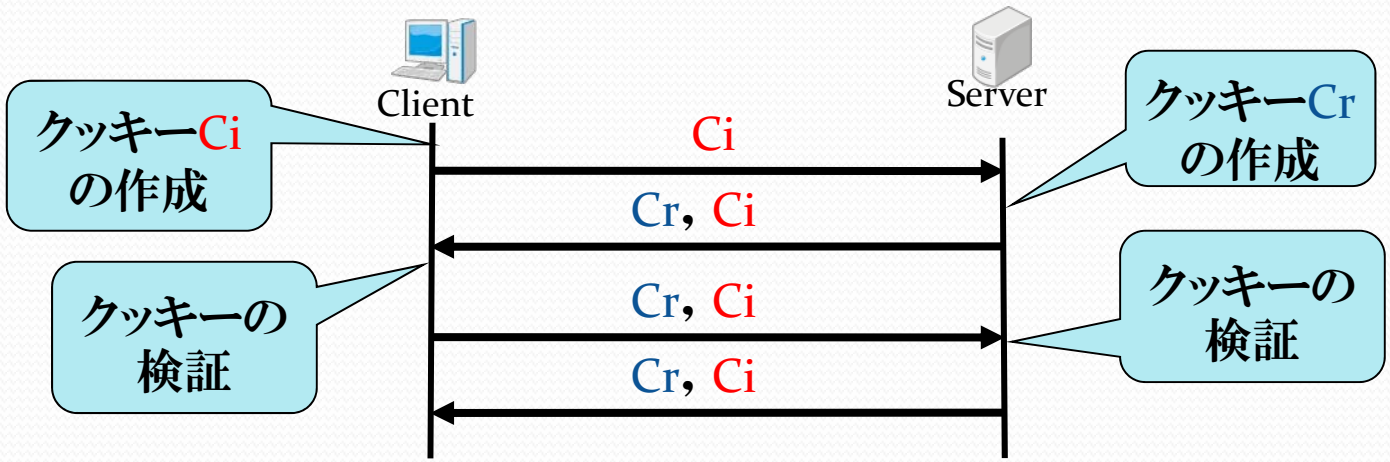
クッキー	ClientのIP
C_A	A
C_B	B
C_C	C

セキュリティ3＜Dos攻撃＞

Cookieで対応

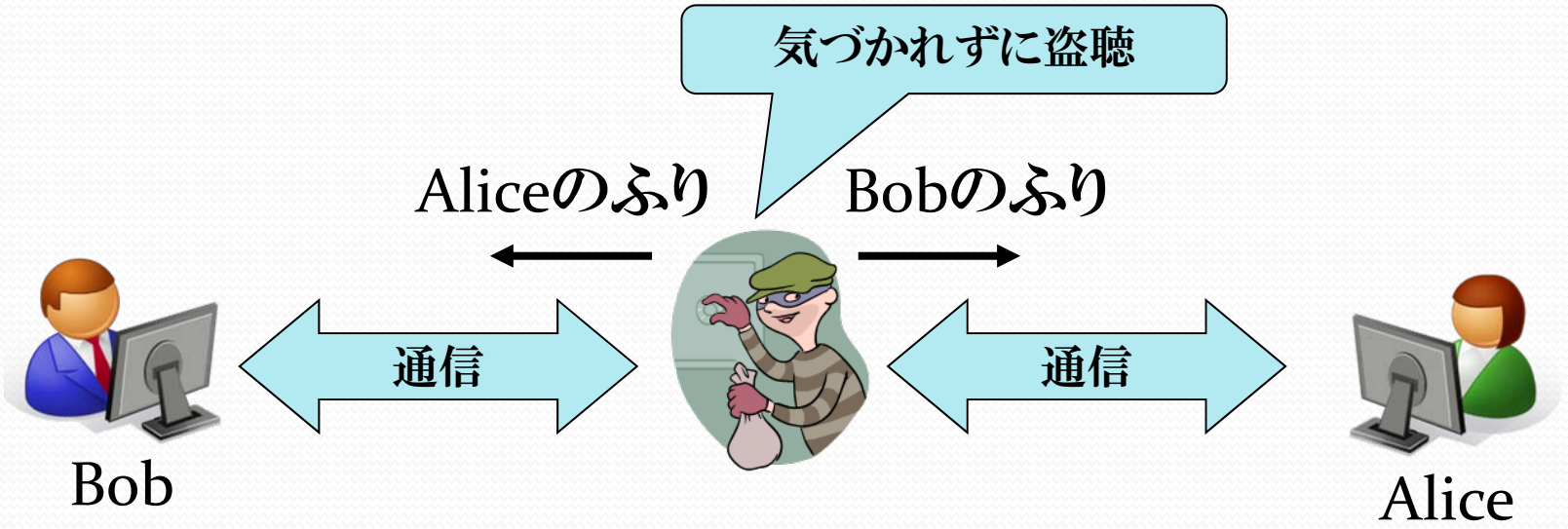
- Dos攻撃
  - サーバに対して大量のデータを送信することで、異常動作やダウンなどを引き起こさせる攻撃
- Cookie（作成日時を含む）
  - 送信元IPアドレスと送信先IPアドレス、乱数を基に作成

⇒無関係なリクエストは拒否できる

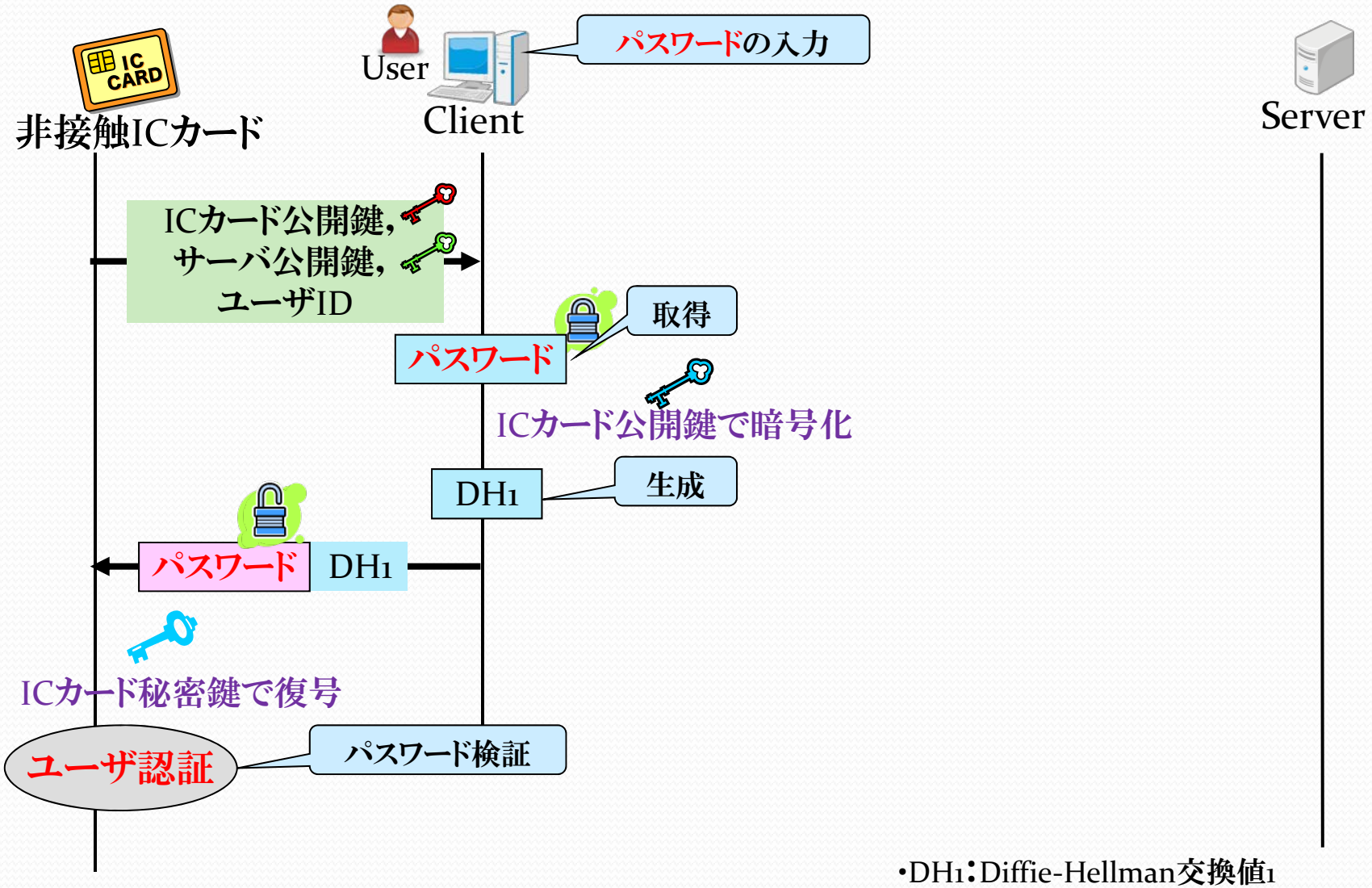


中間者攻撃

- 通信を行う二者間に割り込んで、両者が交換し合う情報を攻撃者の情報にすりかえる手法
  - 通信相手が正当な相手か確認することが必要

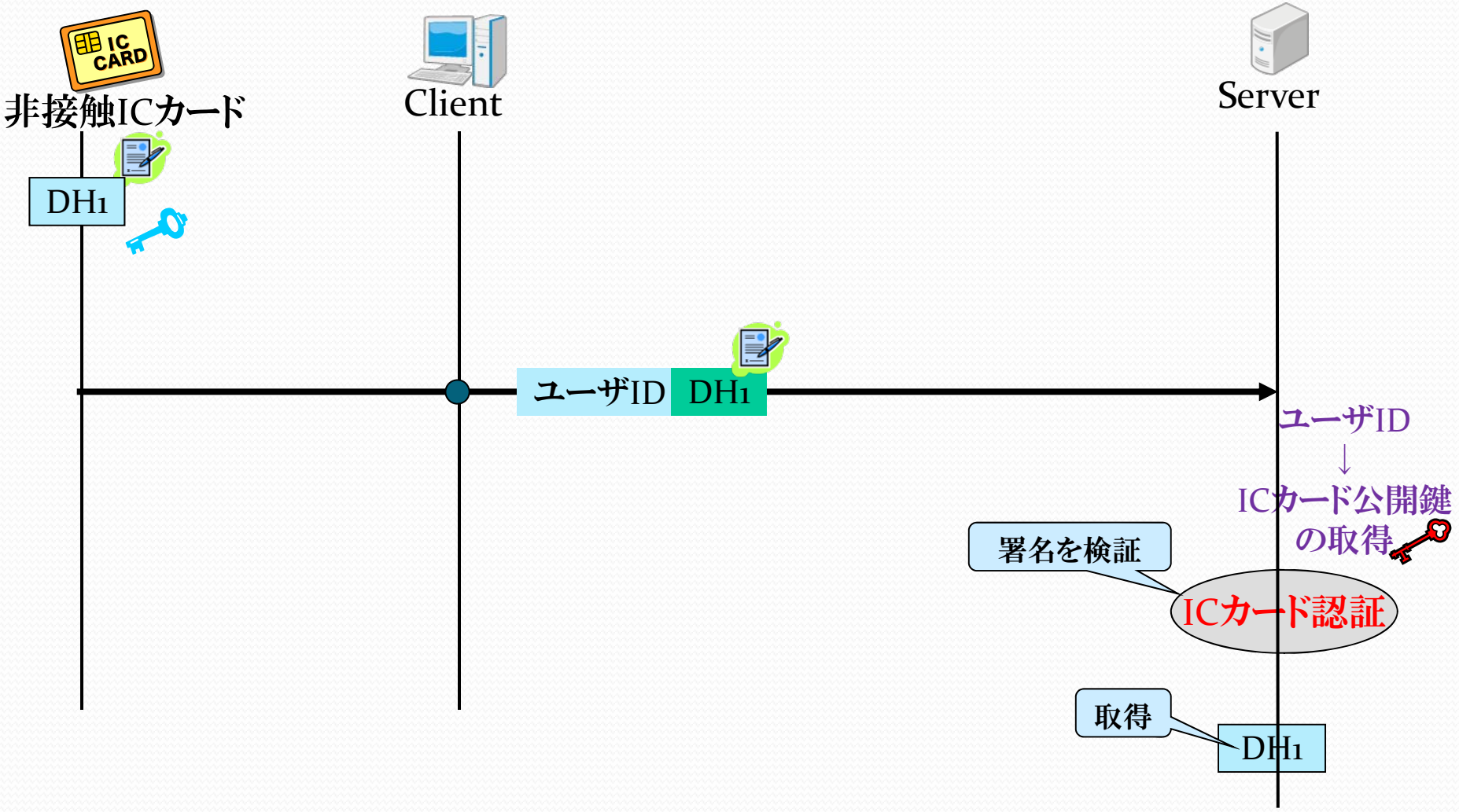


SPAICの動作1＜ユーザ認証＞

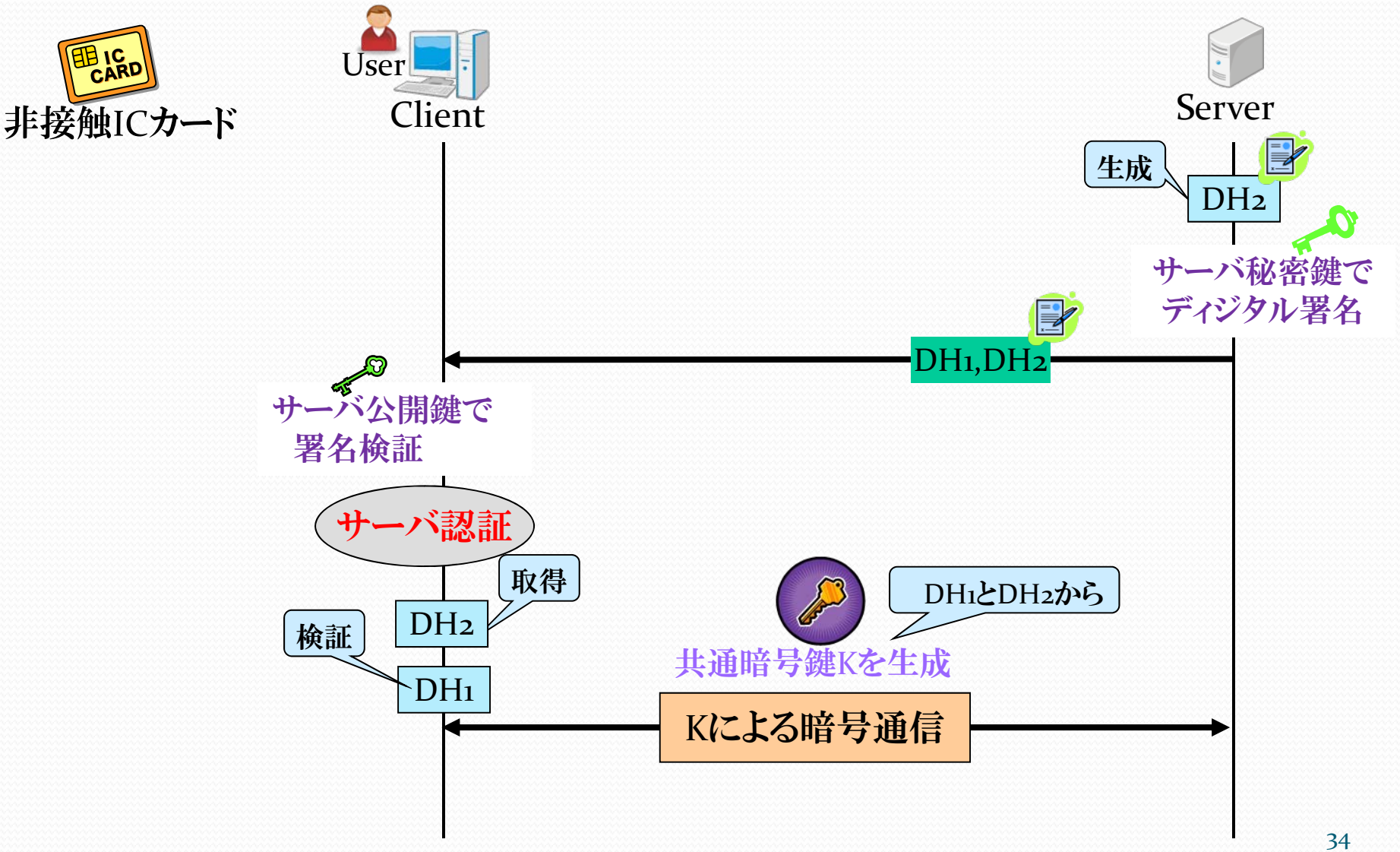


•DH1：Diffie-Hellman交換値1

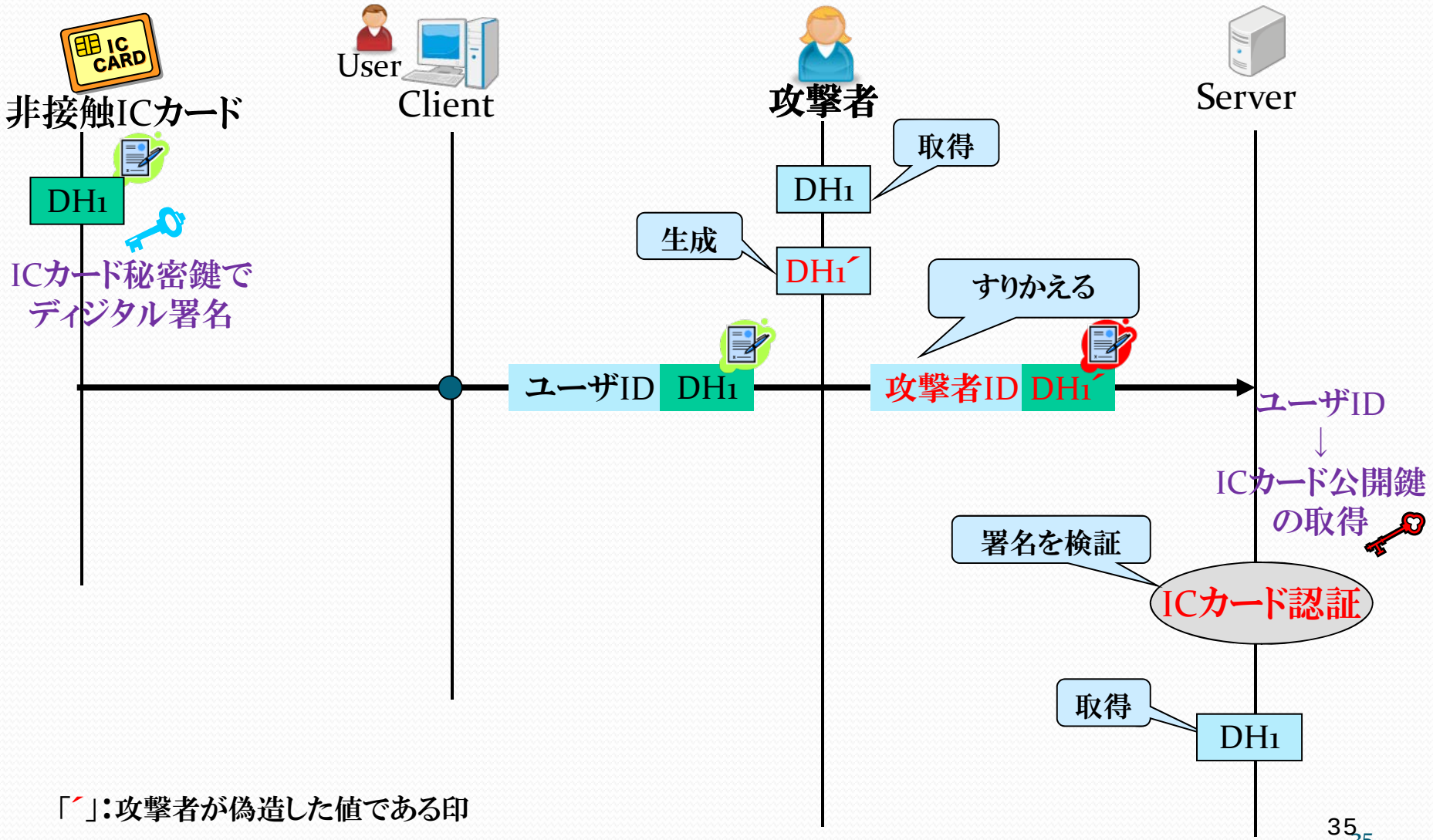
SPAICの動作2＜ICカード認証＞



SPAICの動作3＜サーバ認証＞



中間者の動作1＜ICカード認証＞



「´」：攻撃者が偽造した値である印

35

中間者の動作2＜サーバ認証＞