Anonymous and Transparent Gateway-based Authenticated Key-Exchange

Michel Abdalla, Malika Izabachène et David Pointcheval

École Normale Supérieure (Paris)

23 juillet 2009

Key Exchange Protocol

A fundamental problem in cryptography: enable secure communication over insecure channels

Key Exchange Protocol

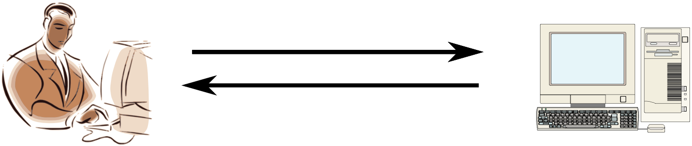
A fundamental problem in cryptography: enable secure communication over insecure channels

A common scenario: Users encrypt and authenticate their messages using a shared secret

Key Exchange Protocol

A fundamental problem in cryptography: enable secure communication over insecure channels

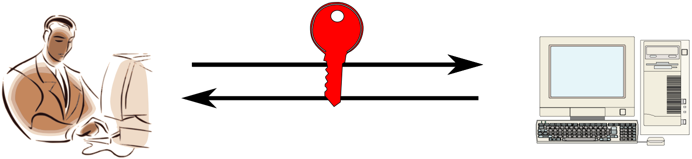
A common scenario: Users encrypt and authenticate their messages using a shared secret



Key Exchange Protocol

A fundamental problem in cryptography: enable secure communication over insecure channels

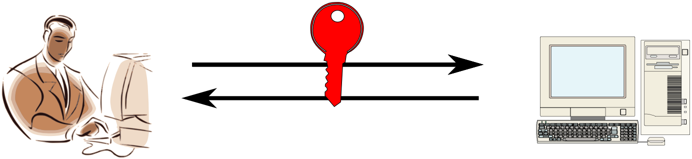
A common scenario: Users encrypt and authenticate their messages using a shared secret



Key Exchange Protocol

A fundamental problem in cryptography: enable secure communication over insecure channels

A common scenario: Users encrypt and authenticate their messages using a shared secret



How to share a secret key?

Key-exchange protocol

Diffie-Hellman protocol [DH76] (1/2)

$\mathbb{G}$ is group where the CDH problem is hard to solve
g a public generator of $\mathbb{G}$

The CDH Problem:
Given g, g^x et g^y, compute g^{xy}
(x and y are private)

The DDH problem:
Given g, g^x, g^y et z, decide whether $z \stackrel{?}{=} g^{xy}$
(x are y private)

Diffie-Hellman Protocol[DH76] (2/2)

$\mathbb{G}$ is a group where the CDH problem is hard to solve
g a public generator of $\mathbb{G}$

Alice		Bob
$sk_A \xleftarrow{R} \{0, \ldots, \lvert\mathbb{G}\rvert - 1\}$		
$pk_A = g^{sk_A}$	$\xrightarrow{pk_A}$	$sk_B \xleftarrow{R} \{0, \ldots, \lvert\mathbb{G}\rvert - 1\}$
	$\xleftarrow{pk_B}$	$pk_B = g^{sk_B}$

The common secret key is $sk = {pk_B}^{sk_A} = g^{sk_A sk_B} = {pk_A}^{sk_B}$

Problem: this scheme is not authenticated

Both Alice and Bob don't know to whom they are actually speaking

Authenticated Key Exchange (AKE)

Allow two parties to establish a common session key
$\longrightarrow$ in an authenticated manner

- Intuitive Goal: provides implicit authentification
  The session key should be known to the involved parties

- Formal Modelisation: provides Semantic Security of the key

  The session key should be indistinguishable from a random string

Forward-secrecy: even if a long-term secret is exposed (in the future), the security of the current session key is preserved.

Diffie-Hellman: a *man-in-the-middle* is possible
$\longrightarrow$ no authentification is possible

Authentification Techniques

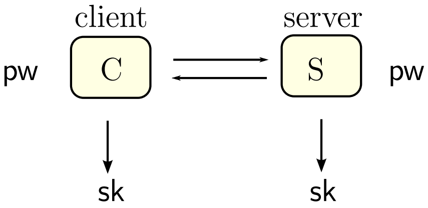
- Asymmetric Techniques
  We assume the existence of a PKI (*public-key infrastructure*).
  Each user owns a pair (secret key, publlic key) given to him by a trusted authority

- Symmetric Techniques
  Users share a random secret key
- Password-Based Techniques
  Users shares a low-entropy secret key
  Example: about 4-digits pincode, a ssh password

⟶ Password-Based authentification

Outline

Password-Based AKE



✓ Realistic:
Real life applications actually rely on weak passsords

✓ Convenient to use:
Users do not need to store the secret

✗ But subject to online dictionary attacks: Unvoidable attacks (small size of the dictionnary)

Online Dictionary Attacks

Let D be the set of all possible passwords (*dictionary*) from which are drawn random passwords $\longrightarrow$ $|D|$ is small

Online dictionary attacks:

- choose a password in D
- interact with the authentification server using pw
- each attempt can succeed with probability $1/|D|$

Protection against these attacks:
limit the number of failed attempts

Aim of the password-based authentification:
restrict the adversary to these attacks only

Outline

Communication Model

The users can have several instances running concurrently (U^i for a user and S^j for a server in the two-party case)

- Each user will be associated to an oracle instance
- The i_{th} instance of player U will be called U^i

Communication can be contolled by the adversary ($\mathcal{A}$):

- Insecure channels: $\mathcal{A}$ can create, forward or cancel messages
- Flows can be modified or dropped by $\mathcal{A}$
- Message transmission is done *via* specific queries to the oracles

Adversary's queries (1/3)

- Execute(U^i, S^j)
  $\longrightarrow \mathcal{A}$ obtains the *transcript* of the execution
  Models passive attacks(*eavesdropping*) on an execution of the protocol between U^i et S^j
- Reveal(U^i)
  $\longrightarrow \mathcal{A}$ obtains the established session key of U^i
  Models a misuse of the session key by U^i
- Send(U^i, m)
  $\longrightarrow \mathcal{A}$ sends the message m to U^i
  Models an active attacks against U^i
- Test(U^i)
  $\longrightarrow \mathcal{A}$ obtains U^i's session key if $b = 0$ or a random session key if $b = 1$
  Models the semantic security of U^i's session key

Adversary's queries (2/2)

Notion of partnering:

- two instances are partnered if they share the same sid
- in the standard model, $sid =$ *transcript* of the session
- the probability that two instances share the same sid is negligible

Freshness:

- a player instance is fresh if it has accepted the session key session and if no Reveal query has been asked to it or its partner
- a Test query is forbidden on a non-fresh instance
- the freshness status allows to remove trivial attacks against semantic security

Notions of Security

Advantage of the adversary:

$$\mathrm{Adv}_P^{ake}(t, q_{reveal}, q_{send}, q_{execute}) = \max_{\mathcal{A}}(2\Pr(Succ) - 1)$$

- Succ is the event for which $\mathcal{A}$ guesses the bit involved in a Test query correctly
- q_{reveal}, q_{send} et q_{exe} are the maximum number of queries $\mathcal{A}$ has done to Reveal, Send and Execute oracle.

A PAKE is secure if: $\mathrm{Adv}_P^{ake} \approx q_{send}/|D| + \mathrm{negl}(k)$

$q_{send}/|D|$: online Dictionary attacks

Outline

Extension to the Three party case

Client

pw

Server

pw

Extension to the Three party case

Client

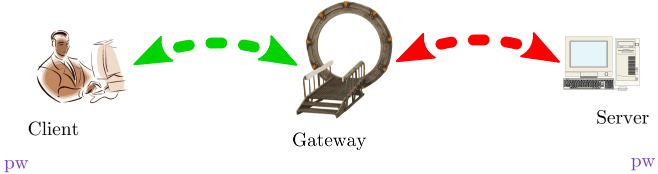
pw

Gateway

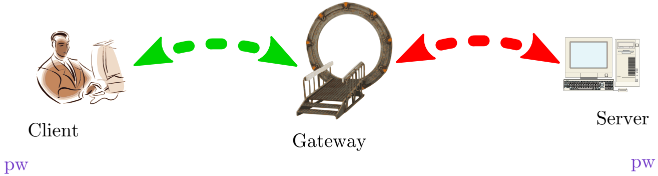
Server

pw

Extension to the Three party case



Extension to the Three party case



Motivation for GPAKE:

- Pratical situation: the authentification task is left to different entites
- Security against off-line Dictionary Attacks is not enough (w.r.t malicious gateway)

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password
2. Key-privacy w.r.t to Server

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password
2. Key-privacy w.r.t to Server
3. Mutual Authentication: the Client and the Gateway are both sure to speak to each other

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password

   Protection against Dictionary attacks w.r.t the Gateway

2. Key-privacy w.r.t to Server

3. Mutual Authentication: the Client and the Gateway are both sure to speak to each other

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password

   Protection against Dictionary attacks w.r.t the Gateway

2. Key-privacy w.r.t to Server

   Guarantee Semantic Security of the session key

3. Mutual Authentication: the Client and the Gateway are both sure to speak to each other

Model for GPAKE

Our goal:

1. the Gateway doesn't learn anything about the password

   Protection against Dictionary attacks w.r.t the Gateway

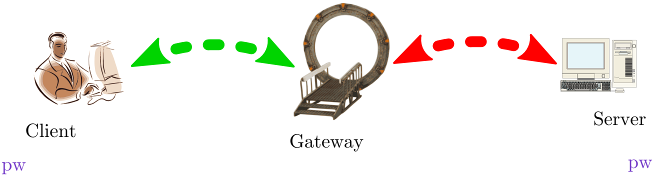
2. Key-privacy w.r.t to Server

   Guarantee Semantic Security of the session key

3. Mutual Authentication: the Client and the Gateway are both sure to speak to each other

   Both compute the real session key with its actual partner

Gateway-Based PAKE [ACFP05]



$\mathsf{PW} = \mathcal{G}(C, S, \mathsf{pw}) \in D \subset \mathbb{G}$ $\xrightarrow{(flow1)}$ $\xleftarrow{(flow4)}$ $\xrightarrow{(flow2)}$ $\xleftarrow{(flow3)}$ $\mathsf{PW} = \mathcal{G}(C, S, \mathsf{pw}) \in D \subset \mathbb{G}$

$\mathsf{sk} = \mathcal{H}_1(\mathsf{transcript})$ $\mathsf{sk} = \mathcal{H}_1(\mathsf{transcript})$

Outline

Our contributions for the model

- A Unified Security Model: consider semantic security and unilateral (resp. mutual) authentification simultaneously
- Stronger Notion of Corruption: even if participants are corrupted (leakage of a long-term scecret), the session can remain *fresh*

  $\longrightarrow$ this allows to consider a stronger notion of perfect forward-secrecy
- Client Anonymity w.r.t the Server: The Server doesn't know which Client is currently connected

  $\longrightarrow$ strengten the Transparency property

More Fresh sessions

Idea: specify the identity of the sender of a Send query

A session is fresh if

1. instances involved have accepeted and
   nobody is corrupted and
   no Reveal query has been asked (as before)

2. all (or some of) messages are oracle generated

$\longrightarrow$ even if a participant is corrupted, the session could be maintened as fresh

Tools for GPAKE

A Diffie-Hellman-Based Assumption: PCDDH

Password chosen basis Diffie-Hellman Assumption

1. Round 1: give D to $\mathcal{A}$
   $\mathcal{A}$ returns a basis X
2. Round 2: choose $s_0, s_1 \overset{\mathcal{R}}{\leftarrow} \mathbb{Z}_q$ and PW $\in D$
   choose $b \in \{0, 1\}$
   set $X' \leftarrow (X/\mathsf{PW})^{s_b}$ and $Y \leftarrow g^{s_1}$
3. Round 3: Given Y and X'
   $\mathcal{A}$ returns his response b' for b

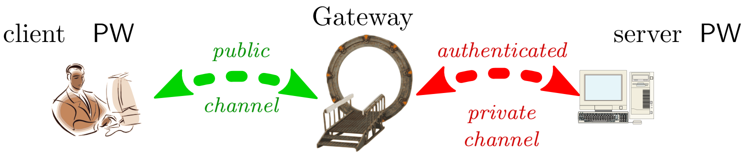
Interactive assumption but ...
quite reasonable assuming DDH holds (if pw is drawn uniformly at random in D)

Gateway-Based PAKE

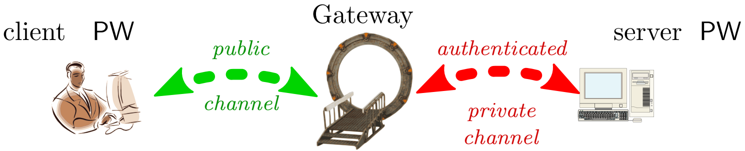
client PW

server PW

Gateway-Based PAKE



Gateway-Based PAKE

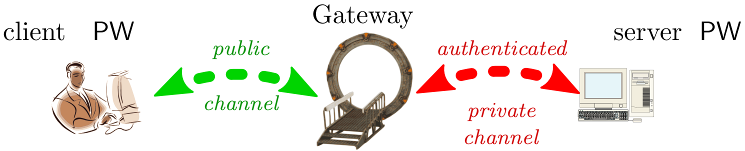


$x \stackrel{R}{\leftarrow} \mathbb{Z}_q$

$X^* \leftarrow g^x \cdot \mathsf{PW}$

$\xrightarrow{\text{C}, X^*}$

Gateway-Based PAKE



$x \xleftarrow{R} \mathbb{Z}_q$

$X^* \leftarrow g^x \cdot \mathsf{PW}$

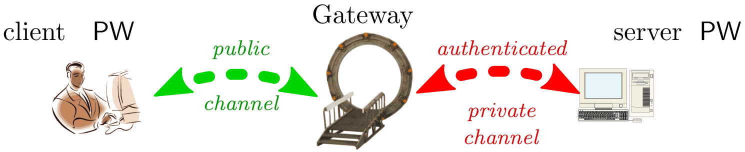
C,X^* →

$y \xleftarrow{R} \mathbb{Z}_q$

$Y \leftarrow g^y$

C,X^*, Y →

Gateway-Based PAKE



Client: $x \overset{R}{\leftarrow} \mathbb{Z}_q$, $X^* \leftarrow g^x \cdot \mathsf{PW}$

$\xrightarrow{\text{C}, X^*}$

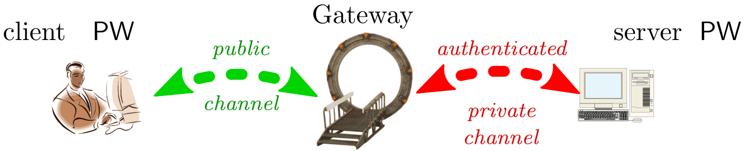
Gateway: $y \overset{R}{\leftarrow} \mathbb{Z}_q$, $Y \leftarrow g^y$

$\xrightarrow{\text{C}, X^*, Y}$

Server: $s \overset{R}{\leftarrow} \mathbb{Z}_q$, $\bar{Y} \leftarrow Y^s$, $\bar{X} \leftarrow (X^*/\mathsf{PW})^s$

$\xleftarrow{\text{S}, \bar{X}, \bar{Y}}$

Gateway-Based PAKE



Client: $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $X^* \leftarrow g^x \cdot \mathsf{PW}$

Client → Gateway: C, X^*

Gateway: $y \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $Y \leftarrow g^y$

Gateway → Server: C, X^*, Y

Server: $s \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $\bar{Y} \leftarrow Y^s$, $\bar{X} \leftarrow (X^*/\mathsf{PW})^s$

Server → Gateway: $\mathrm{S}, \bar{X}, \bar{Y}$

Gateway: $K \leftarrow \bar{X}^y$

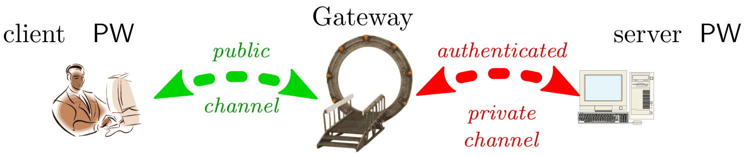
$K \leftarrow \bar{Y}^x$

$\mathsf{AuthG} \leftarrow \mathcal{H}_2(S, G, X^*, \bar{Y}, K)$

Valid(AuthG)?

sk sk

Anonymous Gateway-Based PAKE



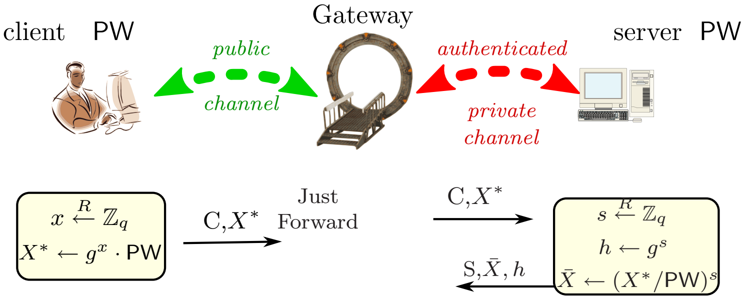
$x \stackrel{R}{\leftarrow} \mathbb{Z}_q$

$X^* \leftarrow g^x \cdot \mathsf{PW}$

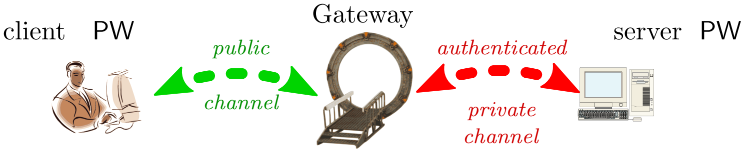
C, X^* $\longrightarrow$ Just Forward

C, X^* $\longrightarrow$

Anonymous Gateway-Based PAKE



Anonymous Gateway-Based PAKE



Client:

$x \xleftarrow{R} \mathbb{Z}_q$

$X^* \leftarrow g^x \cdot \mathsf{PW}$

C, X^* → **Gateway:** Just Forward

C, X^* → **Server:**

$s \xleftarrow{R} \mathbb{Z}_q$

$h \leftarrow g^s$

$\bar{X} \leftarrow (X^*/\mathsf{PW})^s$

S, $\bar{X}, h$ ← **Gateway:**

$y \xleftarrow{R} \mathbb{Z}_q$

$\bar{Y} \leftarrow h^y$

$K \leftarrow \bar{X}^y$

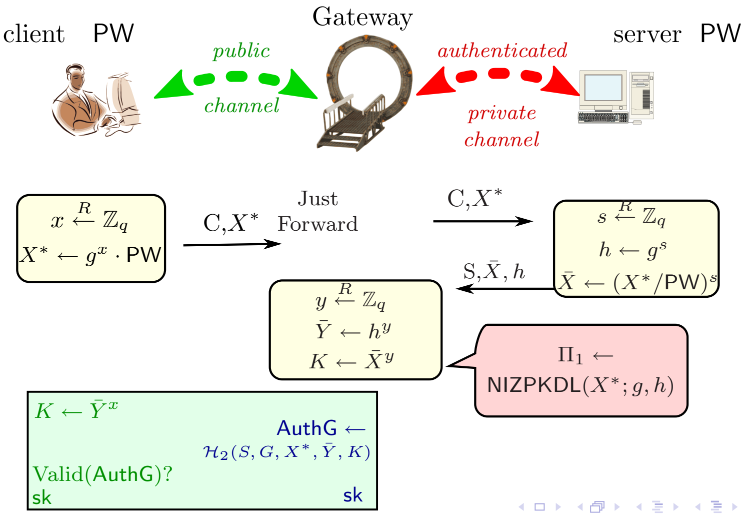
Client: $K \leftarrow \bar{Y}^x$

Gateway: $\mathsf{AuthG} \leftarrow \mathcal{H}_2(S, G, X^*, \bar{Y}, K)$

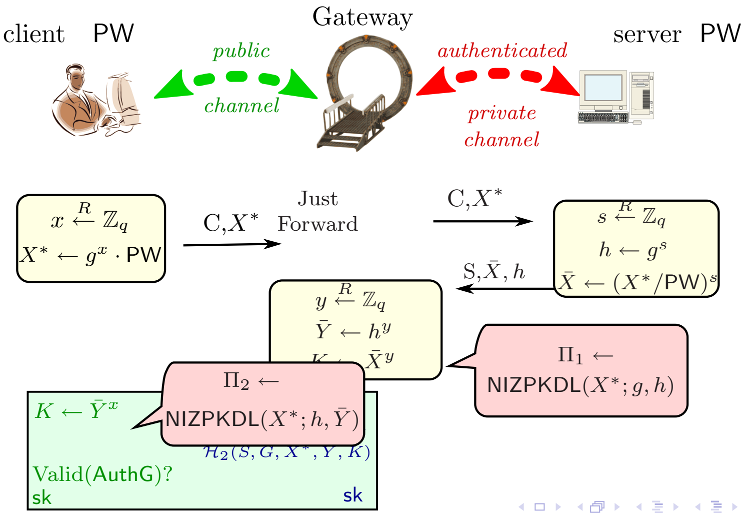
Client: Valid(AuthG)?

sk | sk

Anonymous Gateway-Based PAKE



Anonymous Gateway-Based PAKE



Outline

Key Point of the Proof

General Idea: simulate oracle s.t. $\mathcal{A}$'s view is negligibly close to the one in the real game

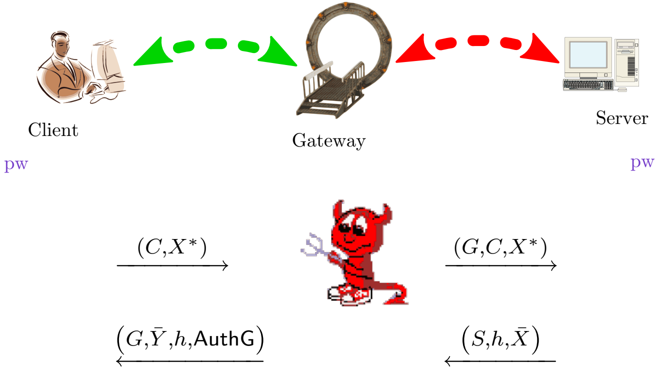
- Nobody is corrupted and $\mathcal{A}$ interacts *passively* (OG or Execute) with the protocol
  $\longrightarrow$ The semantic security relies on the CDH Problem
- If the Gateway is corrupted ?
- If the Client is corrupted ?
- Everybody is corrupted but the all messages are OG?

Trick in the proof (1/4)

More intricate case: the Gateway is corrupted

Trick in the proof (1/4)

More intricate case: the Gateway is corrupted



Trick in the proof (1/4)

More intricate case: the Gateway is corrupted

1. $\mathcal{A}$ asks the help of the server

   But the server oracle still uses PW
   $\longrightarrow$ make sure that Dictionary attacks are not possible

2. $\mathcal{A}$ plays on the behalf of the server and the gateway.

   reject non-oracle generated authenticators
   $\longrightarrow$ compute the probability of bad rejection

Trick in the proof (2/4)

Case 1: $\mathcal{A}$ plays wih a server oracle

$\longrightarrow$ use the PCDDH assumption to reduce $\mathcal{A}$'s task in deciding whether $(g, h, \bar{X})$ is real or random

if $\mathcal{A}$ do so, we have an adversary against the PCDDH Assumption

How?

prove it by an hybrid argument on the number of q_{send} queries to the Server

Trick in the proof (3/4)

Case 2: $\mathcal{A}$ plays on the behalf of the server and the gateway

we assume as symmetric authentication means b.t G and S

$\longrightarrow$ reject non oracle generated authenticators

can we detect it?

YES

Probability of Bad rejection?

Negligible if the client is not corrupted

Trick in the proof (4/4)

- Note that $(g, X^*/\text{PW}, h, \bar{X})$ is a CDH-tuple for at most one PW
- Goal of the simulation: Not use the password anymore

$\longrightarrow$ Goal: show that $P_1 = \frac{qsend}{N}$

Outline

Adding Client Anonymity

The client may want to obtain a session key without letting the server know who he is

$\longrightarrow$ make the Client connections anonymous and unlikable

A Solution

- The Server is viewed as a dynamic database
- For each connection, construct all possible answers for the Client
- The Gateway gets the one for the Client

Adding Client Anonymity/Interface with a PIR

Feature of our GPAKE variant: can be efficiently interfaced with any *Private Information Retrieval*

Adding Client Anonymity/Interface with a PIR

Adding Client Anonymity/Interface with a PIR

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

Adding Client Anonymity/Interface with a PIR

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

requirement: the amount of communication must be less then n.

Adding Client Anonymity/Interface with a PIR

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

requirement: the amount of communication must be less then n.

Two approaches:

Adding Client Anonymity/Interface with a PIR

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

requirement: the amount of communication must be less then n.

Two approaches:

- Information Theoretical: sub-linear communication is not possible for one database [CGKS95].

  Necessity of duplication of databases that do not collude

Adding Client Anonymity/

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

requirement: the amount of communication must be less then n.

Two approaches:

- Information Theoretical: sub-linear communication is not possible for one database [CGKS95].

  Necessity of duplication of databases that do not collude

- Computational: one database is possible [KO97]
  Counterpart: more computational cost

Adding Client Anonymity/

PIR problematic: allow a user to retrieve an item in a database (of size n) without letting know the server which index is asked

requirement: the amount of communication must be less then n.

Two approaches:

- Information Theoretical: sub-linear communication is not possible for one database [CGKS95].

  Necessity of duplication of databases that do not collude

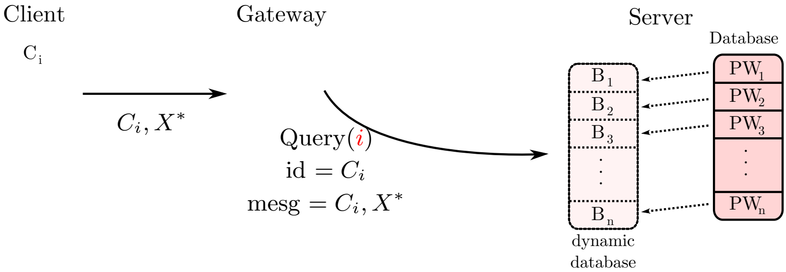
- Computational: one database is possible [KO97]
  Counterpart: more computational cost

Symmetrical PIR: Prevents user from learning more than one item of the database during a session [KO97,GIKM98]

Adding Client Anonymity

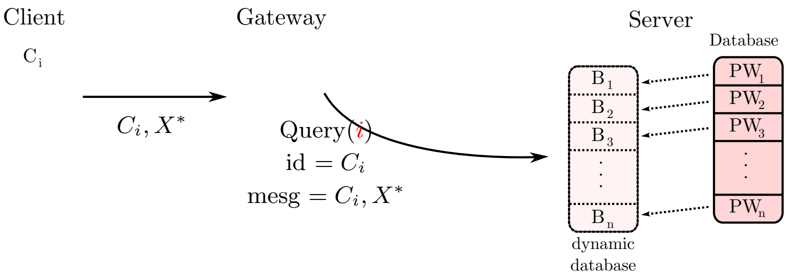
Client		Gateway
$\mathrm{C_i}$		
	$\xrightarrow{C_i, X^*}$	

Adding Client Anonymity



$$B_i = (g^{s_i}, (X^*/\mathsf{PW}_i)^{s_i}, \Pi_i)$$
$$s_i \stackrel{R}{\leftarrow} \mathbb{Z}_q$$

Adding Client Anonymity



Once for all: $h = g^s$, Π_1, $s \stackrel{R}{\leftarrow} \mathbb{Z}_q$

$B_i = (X^*/\mathsf{PW}_i)^s$

- improve computational cost
- improve storage space

Conclusion

- Formalisation of a strenghten model for PAKE protocols considering a broader notion of Freshness
- Apply it to GPAKE
  But still makes sense for the two-party case
  $\longrightarrow$ lead to a (partial) mechanization of the proof (we only consider weak and static corruption)
- Suggest Client Anonymity

Open Question:

- Deal with Dynamic corruption
- Consider other distribution for the Dictionary than a uniform one