Universally Composable Secret Handshakes with Credentials

Nathalie Casati

September 17, 2009

Master Thesis

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
IBM Research - Zurich

People

From EPFL - Laboratory of Cryptologic Algorithms:

- Professor: Arjen Lenstra
- Martijn Stam

From IBM Research - Zurich

- Advisor: Thomas Groß
- Jan Camenisch
- Victor Shoup

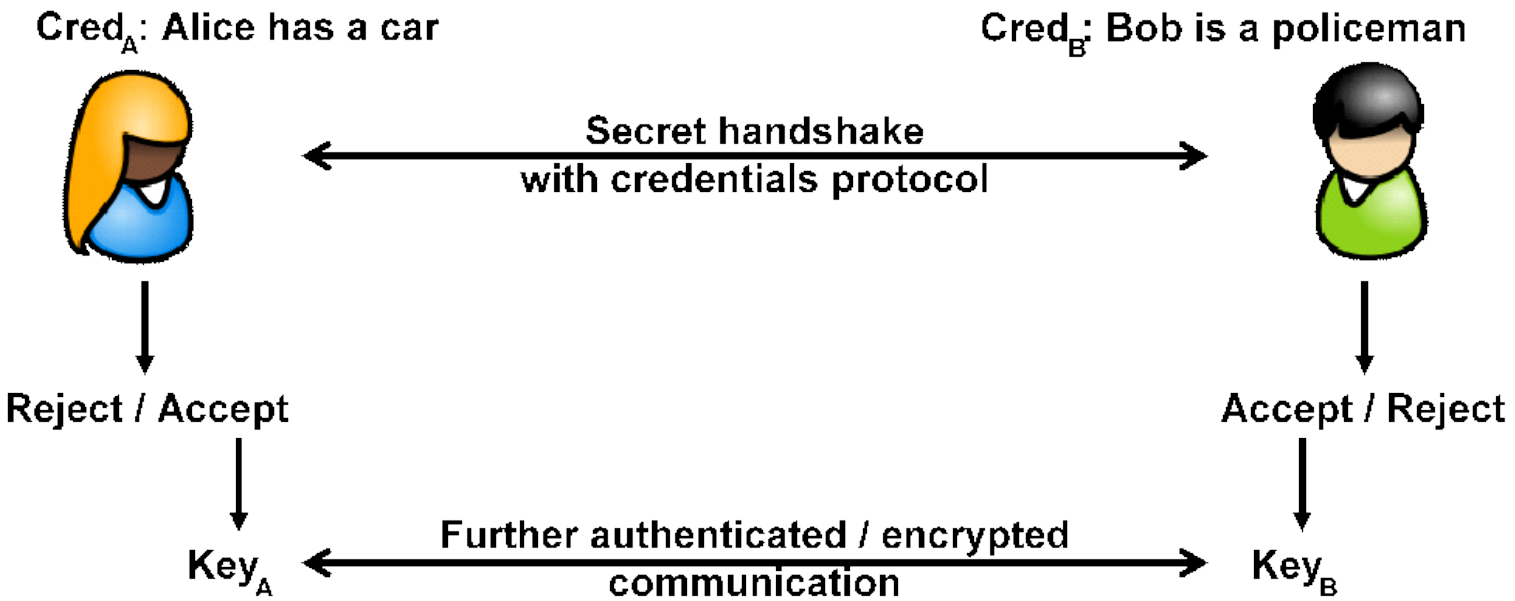
Table of Contents

Table of Contents

Our new problem: Secret Handshakes with Credentials

Credential: a set of attributes including private information signed by an issuer and bound to some party

Secret handshake: a protocol that allows two parties to authenticate each other under specific "conditions". If conditions hold, parties derive a shared common key and can communicate further. Otherwise, parties cannot make any conclusions about the veracity of these "conditions".



The original problem: Secret Handshakes

Conventional Secret Handshakes protocols allow two parties to authenticate each other if and only if they belong to the same group. If they don't, they learn nothing about the group affiliation of the other party.

Application:

- Alice and Bob are secret agents and need to communicate
- They have never seen each other
- Rule of the secret agency: do not reveal your affiliation if the other party is not a secret agent of the agency
- $\longrightarrow$ Neither Alice nor Bob want to show her/his badge first

Secret handshake protocols solve this problem.

Secret Handshakes with Credentials: Practical Application

We can build an online dating service with a list of criteria:

- Eye color
- Age
- Height

All attributes are certified (present on ID cards).

Alice wants to meet Bob if he is taller than 1m78 and between 25 and 30.
Bob wants to meet Alice if she has brown eyes and is younger than 30.

They achieve a match if and only if their credentials fulfill the policy described above, and derive a common shared key. Otherwise they learn nothing about the attributes of the other.

Comparison between Our and Former Schemes

Secret handshake with credentials

- Uses credentials for authentication
- Common shared key is derived
- Policies have to be unrelated
- Issuance is not part of the scheme, instead it can be plugged into IBM Idemix (library implementing an anonymous credential system)

Conventional secret handshake

- Uses group membership for authentication
- Common shared key is derived
- Policies have to be related
- Issuance is part of the scheme

Table of Contents

State of the Art

- Topic introduced in 2003 by Balfanz et al. in *"Secret Handshakes from Pairing-Based Key Agreements"*.
- About 15 other papers on the topic
- Proof of security by reduction to a cryptographic assumption: if the protocol breaks, we can construct an adversary that {breaks RSA, computes discrete logs, ...}
- More features: multi-party handshakes, roles, fuzzy/dynamic matching, multiple groups, ...

Table of Contents

Our Contribution

- The new scheme allows parties to authenticate each other using credentials instead of group membership
- Focus on proving the security in the Universal Composability framework instead of introducing new features

Table of Contents

The Universal Composability (UC) Framework

- Introduced in 2001 by Canetti in *"Universally Composable Security: A New Paradigm for Cryptographic Protocols"*
- Security of protocols maintained under composition with other protocols...
- ...even when running concurrently with any number of arbitrary protocols controlled by the adversary
- Allows modular design of complex schemes by splitting them into simpler sub-protocols
- A number of cryptographic primitives have already been proven secure in UC: digital signatures, encryption, zero-knowledge, commitment, key-exchange, password authenticated key-exchange, ...
- No secret handshake scheme proven secure in UC

Introduction to the UC framework (1)

- **The main idea**: prove that the studied protocol is at least as secure as an *ideal process* for the same goal
- No attack can take place in the ideal world, where the ideal process is running
- If we can prove that the real world process (our protocol) emulates the ideal process, then the protocol has the same security as the ideal process
- In other words: only the attacks possible in the ideal world are possible in the real world

Introduction to the UC framework (2)

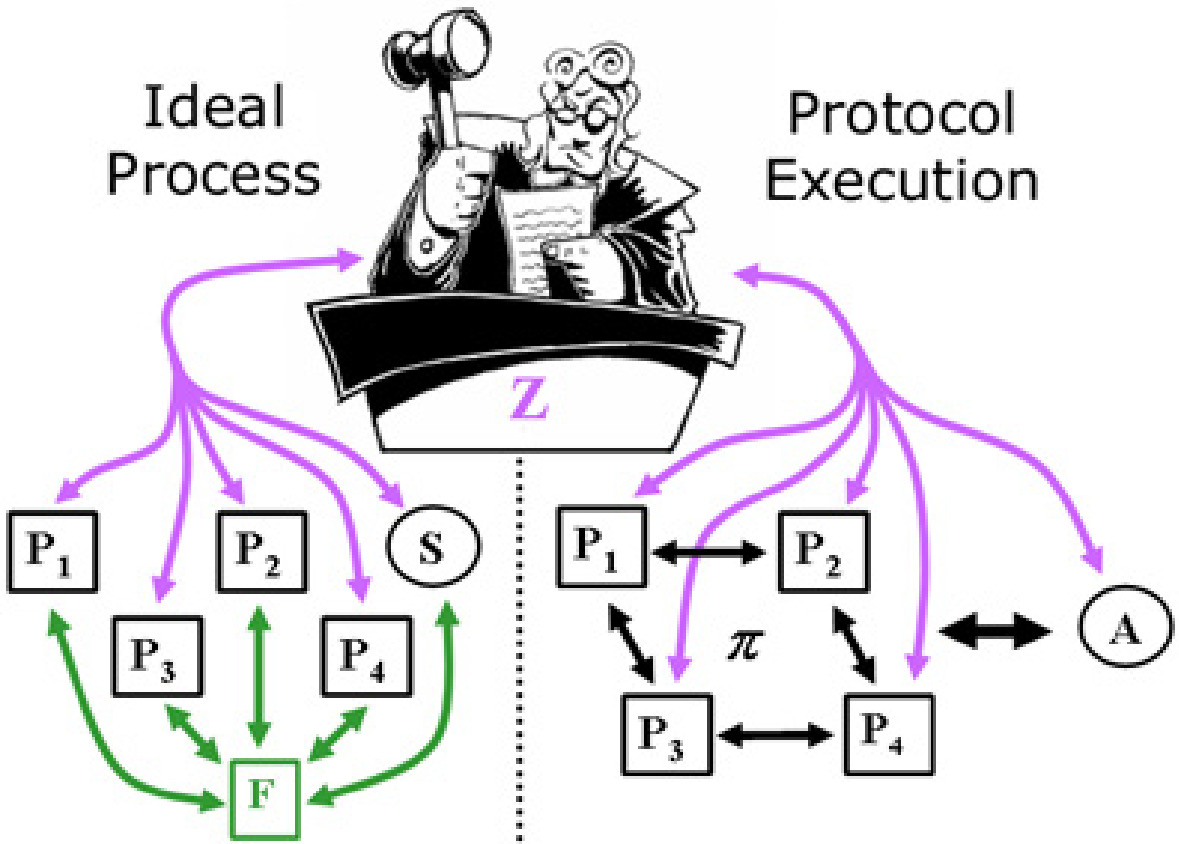
The real world

- Parties get their inputs from the environment $\mathcal{Z}$
- Parties execute the actual protocol Π
- Parties send their outputs to $\mathcal{Z}$
- The real-world adversary is referred to as $\mathcal{A}$

The ideal world

- (Dummy) parties get their inputs from the environment $\mathcal{Z}$
- These inputs are directly forwarded to a trusted party (the ideal functionality) $\mathcal{F}$
- $\mathcal{F}$ computes a predefined function and sends its outputs to the parties, which forward them to $\mathcal{Z}$
- $\mathcal{F}$ is designed so that it is perfectly secure
- The ideal-world adversary is referred to as $\mathcal{S}$

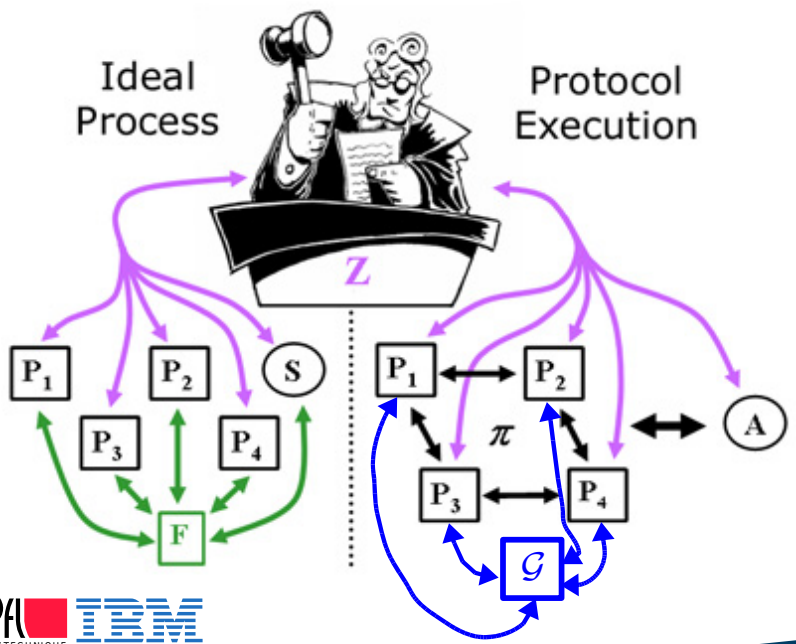
To prove the security of Π, we construct a simulator $\mathcal{S}$ (the ideal-world adversary) having access to $\mathcal{A}$ as a black box, and show that $\mathcal{Z}$ cannot distinguish between an interaction of $\mathcal{A}$ and Π from an interaction of $\mathcal{S}$ and $\mathcal{F}$. In this case we say that Π securely realizes $\mathcal{F}$.



The hybrid Model and the Composition Theorem

The hybrid model:

- Let ρ be a protocol that securely realizes an ideal functionality $\mathcal{G}$
- Let Π be another protocol in which parties make ideal calls to different instances of $\mathcal{G}$
- Π is called a $\mathcal{G}$-*hybrid* protocol that uses both real world communication and instances of an ideal functionality $\mathcal{G}$
- A composed protocol Π^{ρ} is built by executing Π and replacing ideal calls to $\mathcal{G}$ with invocations of ρ



The composition theorem:

- If ρ securely realizes $\mathcal{G}$, the protocol Π^{ρ} emulates an execution of Π in the $\mathcal{G}$-*hybrid* model, meaning $\mathcal{Z}$ cannot distinguish between them

The Generalized Universal Composability (GUC) framework

- Unfortunately, the UC framework fails to guarantee composable security for protocols that access some trusted shared setup information
- New notion of security of GUC: a protocol Π that securely realizes an ideal functionality $\mathcal{F}$ in the GUC framework using some global setup does not affect the security of any other protocols more than $\mathcal{F}$ does, even if protocols running concurrently with Π are maliciously constructed and all protocols use the same global setup
- In practice, protocols share state information in a very restricted way, which can be modeled by letting them access only one shared functionality

Some Definitions

- A **MAC** or Message Authentication Code is a value computed by the sender from a key and a message in order to authenticate the latter. Using the same key and the received message, the receiver can recompute the MAC value and check it against the one he received from the sender. If the message has not been altered, the MAC values are identical. Otherwise they differ.
- A Key Derivation Function is a function that takes as input a secret value (in our case it will be a group element) and outputs a key of fixed length. It uses a deterministic algorithm. We will refer to this type of function as **KDF(·)**.
- The Decisional Diffie-Hellman (**DDH**) problem is defined as follows. Let $\mathbb{G}$ be a cyclic group of some large prime order q. Given $T = g^t, D = g^d, R = g^r \in \mathbb{G}$ for random t, d, r in $\mathbb{Z}_q$, output 1 if $td = r \mod q$, 0 otherwise. We say that an adversary solves the DDH problem if it outputs the correct answer with non-negligible probability.

Zero-Knowledge proofs

A zero-knowledge proof is an interactive protocol for one party (the prover $\mathcal{P}$) to prove to another (the verifier $\mathcal{V}$) that a statement is true, without revealing anything other than the veracity of the statement.

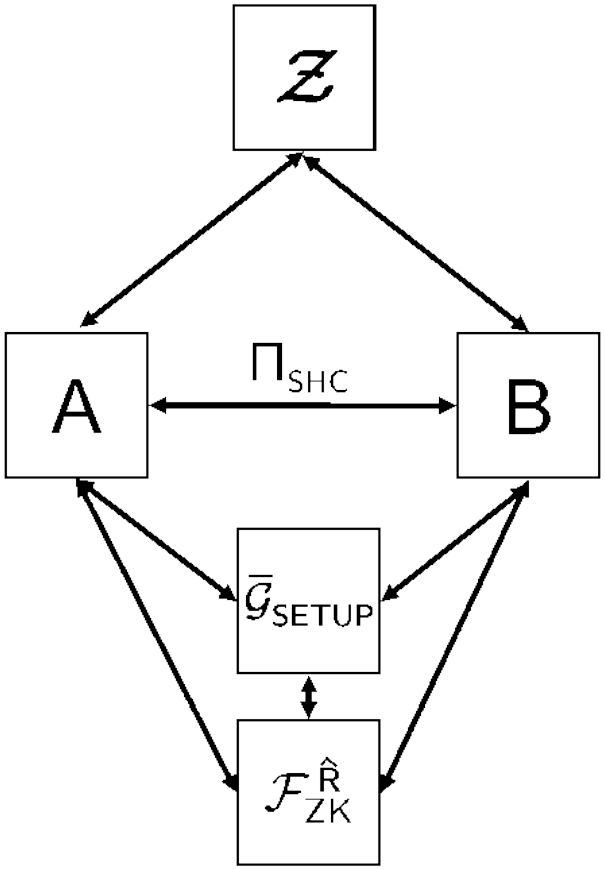
Several zero-knowledge proofs are possible:

- AND proofs: proving that $\mathcal{P}$ is Swiss and has a car
- OR proofs: proving that $\mathcal{P}$ is Swiss or has a car
- Proofs of equality: proving that $\mathcal{P}$'s license plate $=$ ZH 65365
- Proofs that a value lies in a given interval: proving that $\mathcal{P}$ is between 18 and 30

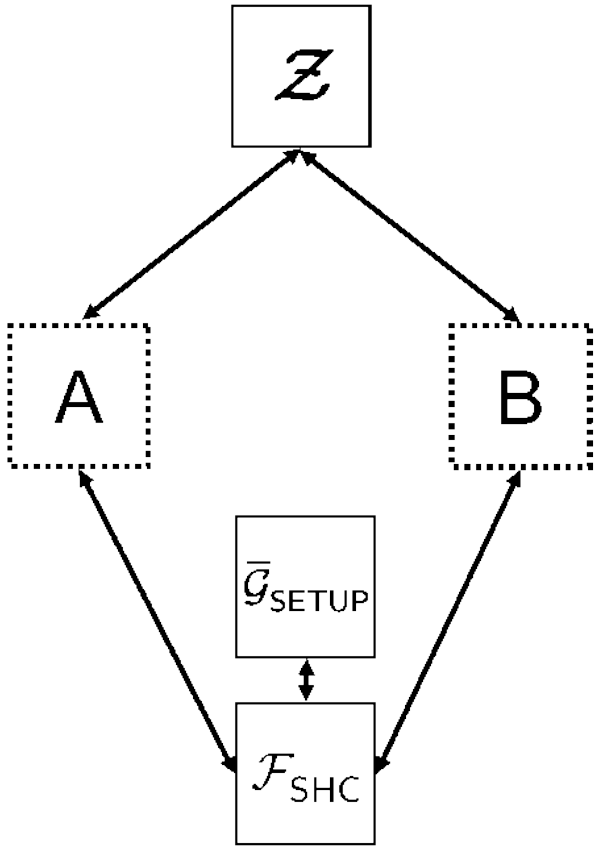
All of this without $\mathcal{P}$ showing his ID or vehicle registration certificate.

Table of Contents

Real and Ideal World: The Big Picture



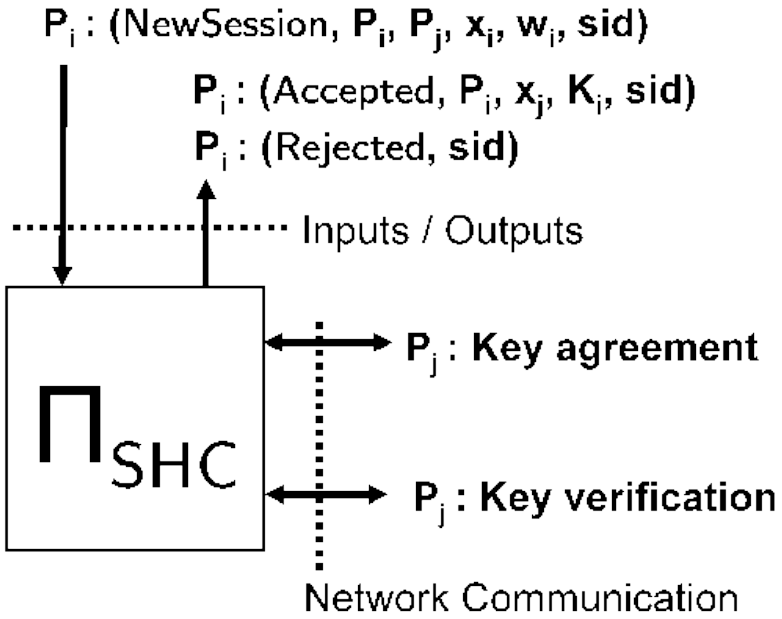
The hybrid world.



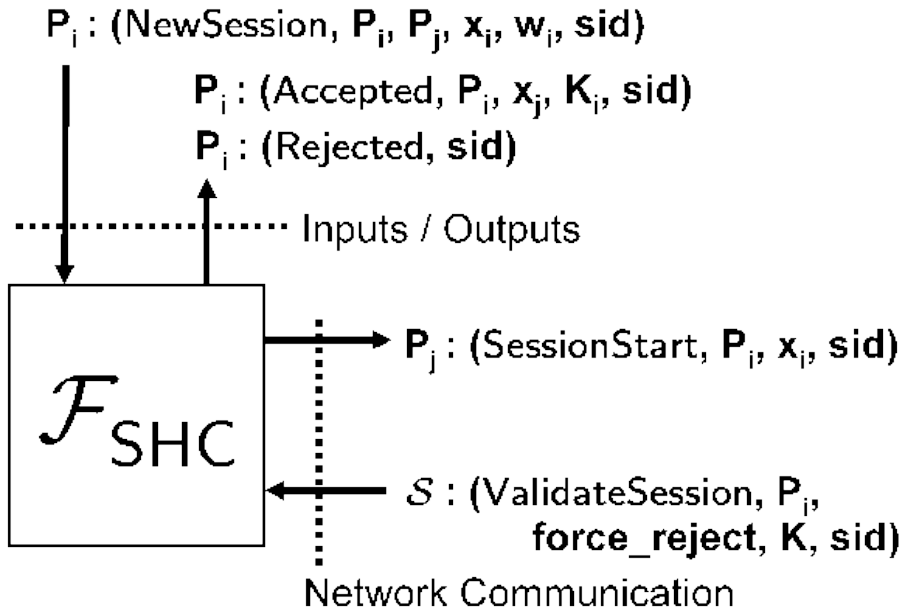
The ideal world.

Service Interface of Π_{SHC} and $\mathcal{F}_{SHC}$: the view of $\mathcal{Z}$

The protocol interface:



The ideal functionality interface:



Service Interface of $\mathcal{F}_{\text{SHC}}$ and Π_{SHC}: the view of $\mathcal{Z}$

Incoming Queries:

- Query (NewSession, P_i, P_j, x_i, w_i, sid) from party P_i:
  Party P_i sends this query to notify P_j that it wants to start a new secret handshake session with it, using the statement x_i and witness w_i. The Session Identifier is set to sid.

Outgoing Queries:

- Private delayed query (Rejected, sid) to party P_i:
  This query is privately sent to P_i and notifies it that its secret handshake session with Session Identifier sid has been rejected.
- Private delayed query (Accepted, P_i, x_j, K_i, sid) to party P_i:
  This query is privately sent to P_i and notifies it that its secret handshake session with Session Identifier sid has been accepted and that its shared key is K_i, and confirms that the other party's witness is x_j.

Queries Specific to the Ideal Functionality $\mathcal{F}_{\text{SHC}}$

Incoming Queries:

- Query (ValidateSession, P_i, *force_reject*, K, *sid*) from $\mathcal{S}$:
  $\mathcal{S}$ sends this query to notify $\mathcal{F}_{\text{SHC}}$ that it is allowed to validate P_i's secret handshake session with Session Identifier *sid*. $\mathcal{F}_{\text{SHC}}$ will output either an (Accepted, ...) or a (Rejected, ...) query to P_i. If P_i or P_j is corrupt, $\mathcal{F}_{\text{SHC}}$ chooses K to be P_i's shared key, which is input by $\mathcal{S}$, otherwise it chooses a random key.

Outgoing Queries:

- Public delayed query (SessionStart, P_i, x_i, *sid*) to party P_j:
  This query is sent to P_j as a consequence of a (NewSession, ...) query from P_i. It will notify P_j that P_i has started a new secret handshake session with it with Session Identifier *sid*.

Table of Contents

The Secret Handshakes with Credentials Ideal Functionality $\mathcal{F}_{\text{SHC}}$

Upon receipt of a query (NewSession, P_i, P_j, x_i, w_i, *sid*) **from party** P_i**:**

- Send public delayed (SessionStart, P_i, x_i, *sid*) to P_j.
- Record (P_i, P_j, x_i, w_i, *sid*) if
  - This is the 1^{st} (NewSession, ...) for this sid
  - This is the 2^{nd} (NewSession, ...) for this sid and $\exists$ a matching record (P_j, P_i, x_j, w_j, *sid*)

Upon receipt of a query (ValidateSession, P_i, *force_reject*, K, *sid*) **from** $\mathcal{S}$**:**

If $\exists(P_i, P_j, x_i, w_i, sid)$, and this is the 1^{st} (ValidateSession, ...) for P_i with this *sid*:

- If P_j is corrupt and *force_reject*=1, send private delayed (Rejected, *sid*) to P_i.
- If $\exists(P_j, P_i, x_j, w_j, sid)$ such that both parties provided a valid witness, then:
  - If P_i or P_j is corrupt, set $K_i \leftarrow K$.
  - Else, if a key K_j has already been sent to P_j, set $K_i \leftarrow K_j$, otherwise choose $K_i \in_R \mathcal{K}$.

  Record K_i and send private delayed (Accepted, P_i, x_j, K_i, *sid*) to P_i.
- Otherwise send private delayed (Rejected, *sid*) to P_i.

In any other case, ignore the query.

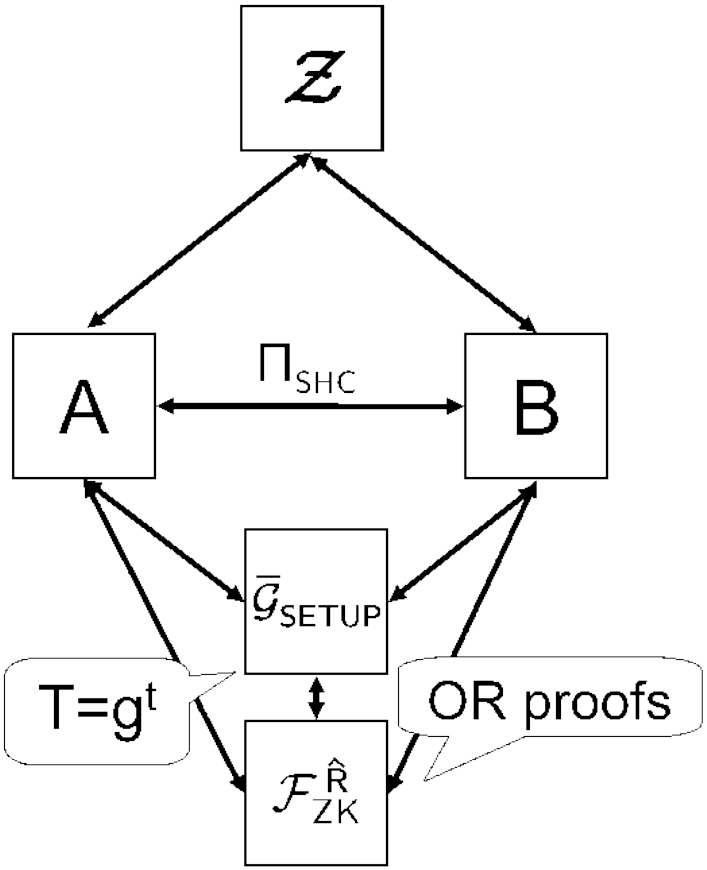
Motivations for $\mathcal{F}_{\text{SHC}}$

- Query (ValidateSession, P_i, *force_reject*, K, *sid*) from $\mathcal{S}$ is introduced to model unfairness: if $\mathcal{A}$ drops messages in the real world, $\mathcal{S}$ can achieve the same result by never sending this query
- Public delayed (SessionStart, P_i, x_i, *sid*) to P_j is introduced to model an imperfection: in the real world $\mathcal{A}$ can see who starts a handshake with whom and with which statement (but $\mathcal{A}$ cannot see if the handshake succeeds)
- Additional parameter *force_reject* is added to (ValidateSession, ...) to model the fact that even if a party B provides a witness at the beginning, he can cheat during the 2^{nd} part of the protocol (using the unfairness), which makes A reject, while B can still accept because he received a valid answer from A. In the ideal world, if *force_reject*=1 and B is dishonest then $\mathcal{S}$ can make A reject

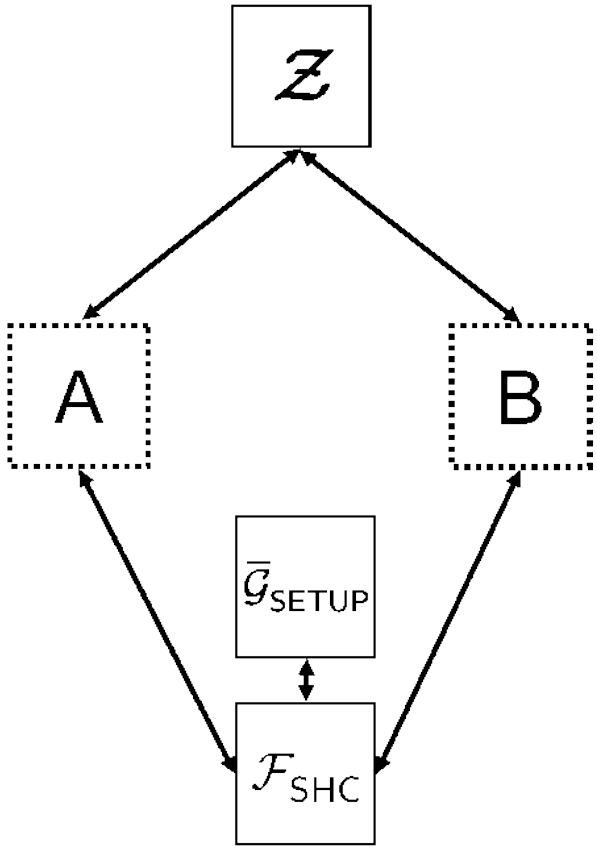
Secret Handshake with Credentials Protocol: The idea (1)

- Party A holds (a credential Cred_A and $value1_A$) or $value2_A$
- Party B holds (a credential Cred_B and $value1_B$) or $value2_B$
- A key is derived from a function of $value1_A$ and $value1_B$
- Algebraic properties ensure that A and B cannot know both values, i.e., A cannot know $value1_A$ *and* $value2_A$, and B cannot know $value1_B$ *and* $value2_B$
- This is to ensure that the shared common session key is possessed by a party if and only if this party owns a valid credential

Zero-Knowledge and Setup Functionalities

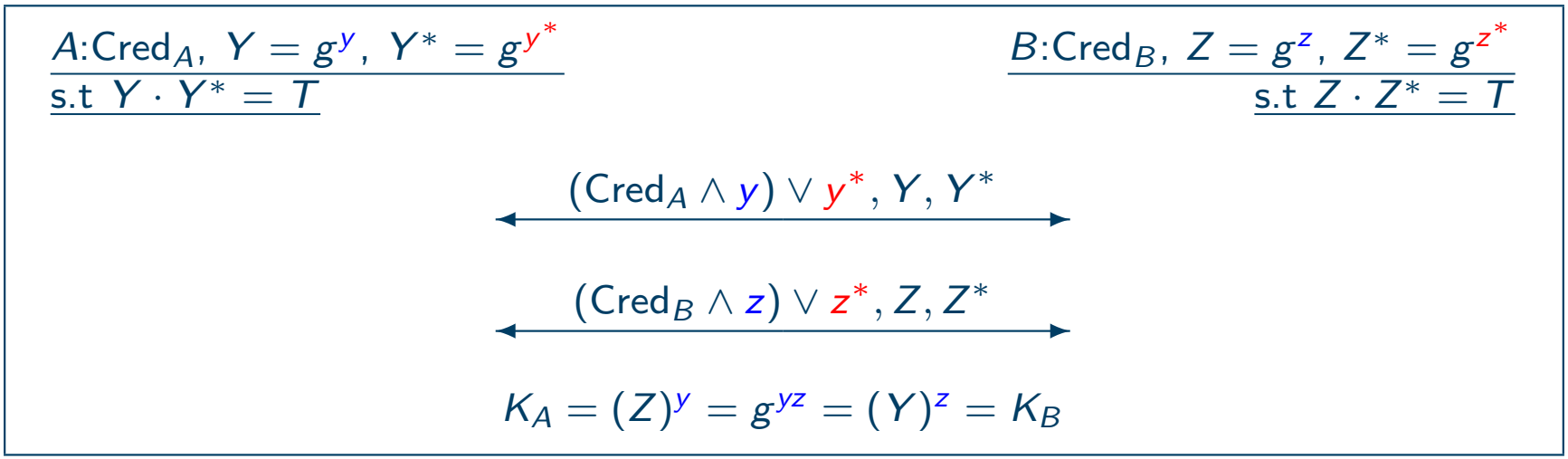


The hybrid world.



The ideal world.

Secret Handshake with Credentials Protocol: The idea (2)



The Secret Handshakes with Credentials Protocol Π_{SHC} (1)

Phase 1: Key agreement

A:(NewSession, A, B, x_A, w_A, sid) $\quad \mathcal{F}_{\mathrm{ZK}}^{\widehat{R}} \quad$ B:(NewSession, B, A, x_B, w_B, sid)

A chooses $y \in_R \mathbb{Z}_q$.
If $w_A = \perp$, then $Y \leftarrow \frac{T}{g^y}$,
else $Y \leftarrow g^y$.

B chooses $z \in_R \mathbb{Z}_q$.
If $w_B = \perp$, then $Z \leftarrow \frac{T}{g^z}$,
else $Z \leftarrow g^z$.

$(\mathsf{Prove}, A, B, (x_A, Y), (w_A, y), sid) \rightarrow$

$\leftarrow (\mathsf{Prove}, B, A, (x_B, Z), (w_B, z), sid)$

$\leftarrow (Ans,\ B,\ A,\ (x_B,\ Z),\ sid)$

$(Ans,\ A,\ B,\ (x_A,\ Y),\ sid) \rightarrow$

where Ans is a variable which may be either
Verified or False for each answer.

The Secret Handshakes with Credentials Protocol Π_{SHC} (2)

Phase 2: Key verification

A:(NewSession, A, B, x_A, w_A, sid)	$\mathcal{F}_{\mathsf{ZK}}^{\widehat{R}}$	B:(NewSession, B, A, x_B, w_B, sid)
If $w_A = \perp$ or $Ans = \mathsf{False}$, then A sets $K_A \in_R \mathcal{K}$. Else if $Ans = \mathsf{Verified}$, $K_A \leftarrow \mathsf{KDF}(Z^y)$. $\mathsf{auth}_A \leftarrow \mathsf{MAC}_{K_A}(sid, A)$		If $w_B = \perp$ or $Ans = \mathsf{False}$, then B sets $K_B \in_R \mathcal{K}$. Else if $Ans = \mathsf{Verified}$, $K_B \leftarrow \mathsf{KDF}(Y^z)$. $\mathsf{auth}_B \leftarrow \mathsf{MAC}_{K_B}(sid, B)$
	$\mathsf{auth}_A \longrightarrow$	
	$\longleftarrow \mathsf{auth}_B$	
If $\mathsf{auth}_B = \mathsf{MAC}_{K_A}(sid, B)$, A outs (Accepted, A, x_B, K_A, sid). Else she outs (Rejected, sid).		If $\mathsf{auth}_A = \mathsf{MAC}_{K_B}(sid, A)$, B outs (Accepted, B, x_A, K_B, sid). Else he outs (Rejected, sid).

The Ideal Functionality for Zero-Knowledge $\mathcal{F}_{\mathrm{ZK}}^{\widehat{R}}$

Upon receipt of a query (Prove, P, V, (x, Y), (w, y), sid) **from prover** P**:**
If $((x, Y), (w, y)) \in \widehat{R}$, send public delayed (Verified, P, V, (x, Y), sid) to V. Otherwise send public delayed (False, P, V, (x, Y), sid) to V.
From now on, ignore future (Prove, ...) queries.

$$\widehat{R} := \{((x, Y), (w, y)) \in L \times \mathbb{G} \times \widehat{W} \times \mathbb{Z}_q | \ ((x, w) \in R \wedge g^y = Y) \vee (w = \perp \wedge g^y = T/Y)\}$$

where

- $\widehat{W} = W \cup \{\perp\}$, the special character $\perp$ meaning that no witness is provided
- $(x, w) \in L \times \widehat{W} \subseteq R$
- L and W are the set of statements and witnesses, respectively
- $(x, w) \in R$ if and only if w is a valid witness for statement x

Table of Contents

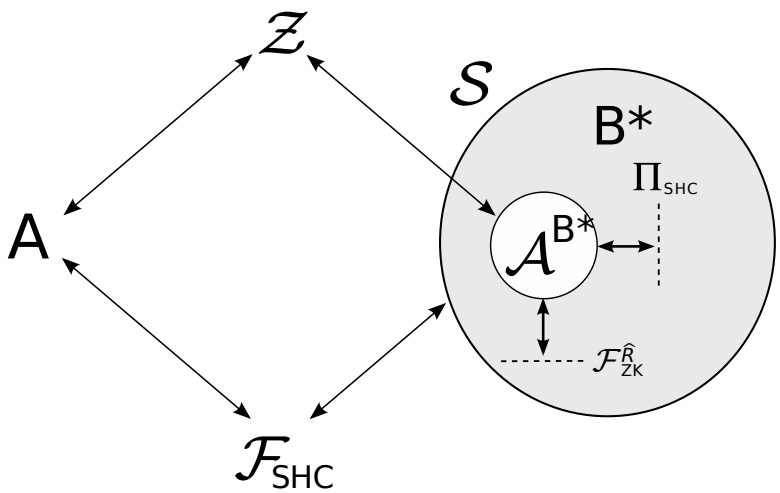
Proof outline (1)

What we have to prove:

- Π securely realizes $\mathcal{F}_{\text{SHC}}$ if for any $\mathcal{A}$ interacting with parties running Π in the $\mathcal{F}_{\text{ZK}}^{\widehat{R}}$-*hybrid* model, there exists a simulator $\mathcal{S}$ s.t. the view of any $\mathcal{Z}$ of an interaction with $\mathcal{A}$ and Π is indistinguishable to $\mathcal{S}$ and $\mathcal{F}_{\text{SHC}}$

How we have to prove it:

- Prove the existence of the simulator $\mathcal{S}$ by constructing it
- Our protocol is symmetric $\longrightarrow$ it is enough to simulate one party
- Let us assume that S plays the role of the corrupted B (notated B^*), hence $\mathcal{S}$ simulates A
- 4 possibilities for A and B^* to have a witness or not

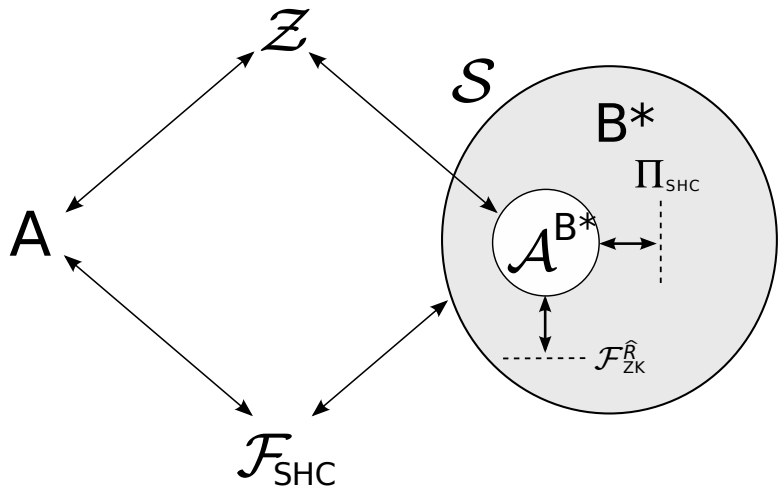


The ideal world.

Proof outline (2)

Ideal world picture:

- $\mathcal{S}$ is simulating A in the execution of Π ($\mathcal{F}_{\mathrm{ZK}}^{\widehat{R}}$-*hybrid* model)
- $\mathcal{S}$ is playing the role of a dishonest party B
- $\mathcal{S}$ simulates $\mathcal{F}_{\mathrm{ZK}}^{\widehat{R}}$

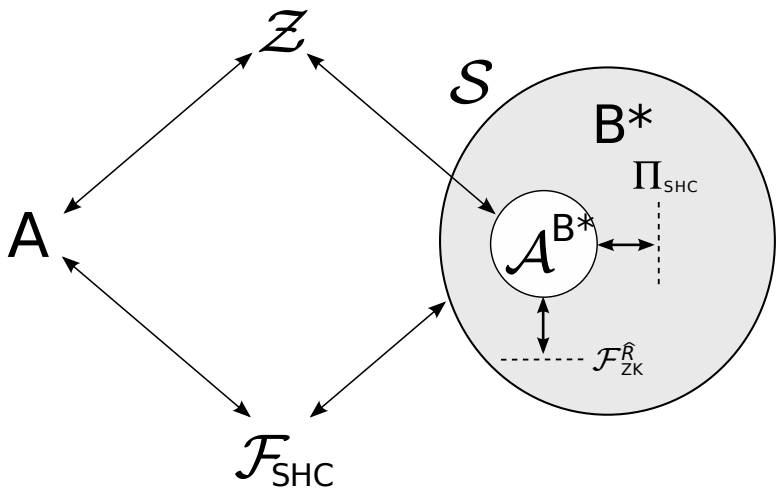


The ideal world.

Proof outline (3)

The main idea:

- $\mathcal{S}$ has to extract knowledge from $\mathcal{A}^{B^*}$ in order to produce the right input for $\mathcal{F}_{\text{SHC}}$
- That means $\mathcal{S}$ has to extract (x_B, w_B) from $\mathcal{A}^{B^*}$ in order to produce the message:
  (NewSession, B^*, A, x_B, w_B, sid)
- To achieve this, $\mathcal{S}$ takes advantage of his knowledge about the ideal world A
- In the case B^* has a witness, $\mathcal{S}$ can guess if A has a witness
- If B^* has no witness, the output is (Rejected, ...). $\mathcal{S}$ uses a random key and we prove that $\mathcal{A}^{B^*}$ cannot see the difference, otherwise we can solve the DDH problem



The ideal world.

Table of Contents

Contribution:

- We have introduced a protocol for secret handshakes which allows matching on credentials
- We have demonstrated that it is secure in the GUC framework using the hybrid model
- We have proposed the first version of an ideal functionality for secret handshakes with credentials

Future work includes:

- Anonymity in the UC framework
- Modeling the issuer
- Realizing the zero-knowledge ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{\widehat{R}}$
- Integrating the policy negotiation step into the protocol

Questions?

The Secret Handshakes with Credentials Ideal Functionality $\mathcal{F}_{\text{SHC}}$

Upon receipt of a query (NewSession, P_i, P_j, x_i, w_i, *sid*) **from party** P_i**:**

- Send public delayed (SessionStart, P_i, x_i, *sid*) to P_j.
- Record (P_i, P_j, x_i, w_i, *sid*) if
  - This is the 1st (NewSession, ...) query for this sid
  - This is the 2nd (NewSession, ...) query for this sid and there is a matching record (P_j, P_i, x_j, w_j, *sid*)

Upon receipt of a query (ValidateSession, P_i, *force_reject*, K, *sid*) **from** $\mathcal{S}$**:**

If there is a record (P_i, P_j, x_i, w_i, *sid*), and this is the first (ValidateSession, ...) query for P_i with this *sid*, then execute one of the following steps:

- If P_j is corrupt and *force_reject*=1, then send private delayed (Rejected, *sid*) to P_i.
- If there is also a record (P_j, P_i, x_j, w_j, *sid*) such that both parties provided a valid witness, execute one of the following steps:
  - If P_i or P_j is corrupt, set $K_i \leftarrow K$.
  - Otherwise, if a key K_j has already been sent to P_j, set $K_i \leftarrow K_j$, otherwise choose $K_i \in_R \mathcal{K}$.

  Then record K_i and send private delayed (Accepted, P_i, x_j, K_i, *sid*) to P_i.
- Otherwise send private delayed (Rejected, *sid*) to P_i.

In any other case, ignore the query.

The Shared Setup Functionality $\bar{\mathcal{G}}_{\text{SETUP}}$

Upon receipt of a query (Setup, *#shsid*) **from any party:**

If this is the first activation, set and record the following parameters:

- $\mathbb{G}$, a group in which the DDH problem is hard, with order q, where q is a large prime
- $g \in \mathbb{G}$, $g \neq 1$
- $T \in_R \mathbb{G}$, where $\log_g(T)$ is an unknown quantity

In any case output (Set, $(\mathbb{G}, g, T)$, *#shsid*) to the activating party.

Solving the DDH problem using $\mathcal{A}^{B^*}$

Let us construct $\mathcal{A}^{\mathrm{DDH}}$ using $\mathcal{A}^{B^*}$:

- Input of $\mathcal{A}^{\mathrm{DDH}}$: T, D, $R \in \mathbb{G}$ (with prime order q)
- B^* has no witness
- $\mathrm{auth}_A \leftarrow \mathrm{MAC}_{K_A}(sid, A)$
- (Verified, A, B^*, (x_A, Y), sid)
- (Prove, B^*, A, (x_B, Z^*), ($w_B = \perp$, z^*), sid)

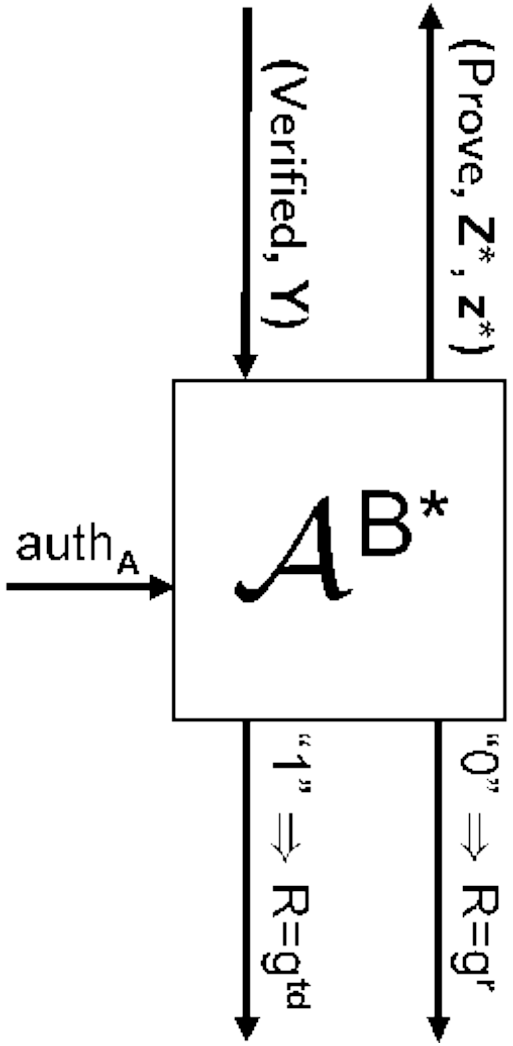
Let us set:

$(g^y =)\ \ Y \leftarrow D$

$(g^z =)\ \ Z \leftarrow T/g^{z^*}$

$K_A \leftarrow \mathrm{KDF}(R/Y^{z^*})$

$\mathcal{A}^{B^*}$ outputs "1" if and only if K_A=KDF(g^{yz}), which means $td = r \mod q$



Camenisch-Lysyanskaya Signature Scheme

Allows one to sign messages $m = (m_0, ..., m_{L-1})$ such that $m_i \in \pm\{0,1\}^{l_m}$.

Key generation:
On input l_n, choose an l_n-bit RSA modulus $n = pq$ such that p and q are safe primes. Choose uniformly at random $R_0, \ldots, R_{L-1}, S$ and $Z \in \mathrm{QR}_n$. Output $(n, R_0, \ldots, R_{L-1}, S, Z)$ as the public key and p as the secret key.

Signing a message $m = (m_0, ..., m_{L-1})$:
Choose a random prime e of length $l_e > l_m + 2$ and a random number v of length $l_v = l_n + l_m + l_r$, where l_r is a security parameter. The signature on the message $(m_0, ..., m_{L-1})$ is (A, e, v), where

$$A = \left(\frac{Z}{R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v} \right)^{1/e} \mod n.$$

Verifying a signature (A, e, v):
Check that $Z \equiv A^e R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v \pmod{n}$, $m_i \in \pm\{0,1\}^{l_m}$ and $2^{l_e} > e > 2^{l_e - 1}$.

The CL signature scheme is secure against adaptive chosen messages attacks under the strong RSA assumption:

It is computationally infeasible, on input of a random RSA modulus n and a random element $u \in \mathbb{Z}_n^*$, to compute values $e > 1$ and v such that $v^e \equiv u \pmod{n}$.