Distant Modeling of NIZK Protocol for Public Keys Certification

Authors:
Alexander Frolov, NRU MPEI
Alexander Vinnikov, Soft Engineering Ltd
Sofya Polyakova, NRU MPEI

Alexander Vinnikov
SoftEngineering Ltd.
We present the IT solutions for educational modeling of non-interactive zero knowledge (NIZK) cryptographic public keys certification protocols. Certified public keys can then be used in a variety of non-interactive protocols, for example, in

• protocols using technique designated verifier proofs,
• authenticated Diffie-Hellman protocol,
• Needham-Schroeder key distribution protocol using asymmetric cryptosystems.

The IT solution is based on algebraic means provided by MPEI algebraic processor.
The IT solution supports the following educational purposes:

- obtaining practical skills of remote implementation of multiple non-interactive oblivious transfer, non-interactive zero-knowledge proof protocols,
- mastering the methods of a remote automatic modeling of cryptographic protocols, non-interactive public keys certification, the technique designated verifier proofs.

The developed educational model allows you to study an interconnected set of cryptographic procedures:

- generation of system parameters of the protocol;
- multiple non-interactive oblivious transfer;
- non-interactive zero-knowledge proof;
- the El Gamal digital signature;
- the Kerberos protocol;
- designated verifier proof
Distance learning is important for studying cryptographic protocols not only as the ability to remotely access information, computing tools and communication, but also as the ability to *simulate protocol execution by remotely separated participants*.

The method of FSM (Finite State Machine) remote modeling of cryptographic protocols was proposed by the authors


It is used in this work to model a non-interactive protocol for public key certification and non-interactive designated verifier proof.
Non-interactivity of the certification protocol is ensured by the use of

- non-interactive multiple oblivious transfer that is organized on the base of key information of m/n (m out of n) fractional oblivious transfer


as proposed in


- non-interactive implementation of multiple challenger of the Schnorr protocol


proposed in


Secret channel used in this protocol is built using the Kerberos protocol

Distant Modeling of NIZK Protocol for Public Keys Certification

Non interactive proofs
\[ y_{ui} = f(x_{ui}). \]

TC-certificate
C-certificate

TC-certificate
C-certificate
U-certificates
Distant Modeling of NIZK Protocol for Public Keys Certification

Non interactive proofs $y_{ui} = f(x_{ui})$.

Non interactive proofs verification

Kerberos

TC public key

TC-certificate
C-certificate
U-certificates
THE DESIGNATED VERIFIER PROOF PROTOCOL IN ITS FSM MODEL

The designated verifier proof protocol follows the last step of the previous protocol, fulfilled for two users: P (prover) and V (verifier), i.e. after broadcasting of their public keys certificates \( P_{\text{certificate}} \) and \( V_{\text{certificate}} \) in the format of \( U_{\text{certificate}} \). Two steps of the designated verifier proof protocol


are the following

(they are the seventh and eighth steps of the combined protocol).

Step 7. Verification of the verifier and prover public key certificates \( P_{\text{certificate}} \) and \( V_{\text{certificate}} \), computing and broadcasting the non-interactive designated verifier proof.

Step 8. Verification of the verifier and prover public key certificates \( P_{\text{certificate}} \) and \( V_{\text{certificate}} \), getting and verification of the non-interactive designated verifier proof.
The following programs of the protocol model are used: S-program is an initializing program; TC-program is the trusted center program; C-program is the child center program; U-program is the user program. S, TC, and C – programs are uploaded on the separate computers, programs U are uploaded on the separate user computers (we assume two users).
Predicate

`hasattr(share,"<name>")`

has a value True, if `<name>` is published (in this case `<name>` is present on all computers) and the value False otherwise; the line

`share.<name>=<data>`

means the publication of `<name>` (assigning the value True to above predicate);

`del share.<name>`

means assigning the value False to that predicate;

`share.<name>=0`

`del share.<name>`

denotes initialization of the above predicate with initial value False. The lines

`share.<name>=0` below are assumed by default and not represented. The secret channel communication is done in a similar way, but the data is encrypted using the wrap function of the channel context

`share.<name>=context.wrap(<data>, True)`

and decrypted using unwrap function of the context.
## FSM Model of NIZK Public Key Certification Protocol

<table>
<thead>
<tr>
<th>Logical condition of protocol step</th>
<th>Program and protocol step number</th>
<th>Protocol step correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasattr (share, &quot;TC_certificate&quot;) and not hasattr(share, “C_publickey”)</td>
<td>TC, 1</td>
<td>share: TC_certificate = TC_certificate, del share.User1proof, del share.User2proof, del share.User1certificate, del share.User2certificate, del share.user1predproof, del share.user2predproof, del share.client_tok, del share.server_tok, del client_ctx, del server_ctx</td>
</tr>
<tr>
<td>hasattr (share, &quot;TC_certificate&quot;) and not hasattr (share, &quot;C_publickey&quot;)</td>
<td>C, 2</td>
<td>share: C_public_key = C_public_key</td>
</tr>
<tr>
<td>not hasattr (share, &quot;C_certificate&quot;)</td>
<td>TC, 3</td>
<td>share: C_certificate = C_certificate</td>
</tr>
<tr>
<td>hasattr (share, “C_certificate”) and not hasattr (share, “thisuser_proof”)</td>
<td>U, 4</td>
<td>share: thisuser_proof = thisuser_proof</td>
</tr>
<tr>
<td>hasattr(share,” thisuser_proof”)</td>
<td>C, 5</td>
<td>share: thisuser_preproof = thisuser_preproof, del share. thisuser_preproof</td>
</tr>
<tr>
<td>hasattr (share, &quot;thisuser_preproof&quot;)</td>
<td>TC, 6</td>
<td>del share. thisuser_preproof, share. thisuser_certificate</td>
</tr>
</tbody>
</table>
**THE DESIGNATED VERIFIER PROOF PROTOCOL IN ITS FSM MODEL (IN DETAIL. P.7)**

The prover gets the prover and verifier certificates:

\[ P_{\text{certificate}}=\text{share}.P_{\text{certificate}}, \]
\[ V_{\text{certificate}}=\text{share}.V_{\text{certificate}}. \]

and verifies the TC digital signature on them, then gets public keys \( y_{\text{verifier}} \) from \( V_{\text{certificate}} \), generates random elements \( w, r, k \in G \), computes \( TC(w, r, y_{\text{verifier}}) \leftarrow g^w \)

\( y_{\text{verifier}} \pmod{p} \), computes the non-interactive designated verifier proof

\( V_{\text{designated proof}} = (w, r, Commit, Challenge, Response) \) (8),

where

\[ Commit=g^k \pmod{p}, \]
\[ Challenge= h(TC(w, r, y_{\text{verifier}}) \parallel Commit \parallel [M]), \]
\[ Response= k+x_{\text{prover}} \cdot (Challenge+w)(\pmod{q}), \]

where \( x_{\text{prover}} \) is the prover’s secret key, \( h \) is a hashing function.
8. The verifier gets the prover and verifier certificates:

\[ P\_certificate = \text{share}.P\_certificate, \]
\[ V\_certificate = \text{share}.V\_certificate \]

verifies the TC digital signature on them, gets public keys \( y_{\text{verifier}} \) from \( V\_certificate \), and \( y_{\text{prover}} \) from \( P\_certificate \), \( w, r, \) Commit, Challenge, Response form \( \text{share}.V\_\text{designated}\_\text{proof} \), and computes

\[ TC(w, r, y_{\text{verifier}}) \leftarrow g^w y_{\text{verifier}}^r \pmod{p}, \]
\[ \text{Challenge} = h(TC(w, r, y_{\text{verifier}}) \parallel \text{Commit} \parallel [M]). \]

Finally, prover verifies the predicate

\[ g^{\text{Response}} = \text{Commit} y_{\text{prover}}^{(\text{Challenge})} y_{\text{prover}}^w \pmod{p}. \]
**Distant Modeling of NIZK Protocol for Public Keys Certification**

- **FSM Model of the Designated Verifier Proof Protocol**

<table>
<thead>
<tr>
<th>Logical condition of protocol step</th>
<th>Program and protocol step number</th>
<th>Protocol step correction</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HASATTR(SHARE, &quot;VERIFIER_PUBLIC_KEY&quot;) AND HASATTR(SHARE, &quot;PROVER_PUBLIC_KEY&quot;)</code></td>
<td>P,7</td>
<td><code>SHARE.PROVER_PROOF = PROVER_PROOF</code></td>
</tr>
<tr>
<td><code>HASATTR(SHARE, &quot;VERIFIER_PUBLIC_KEY&quot;) AND HASATTR(SHARE, &quot;PROVER_PROOF&quot;)</code></td>
<td>V,8</td>
<td><code>DEL SHARE.PROVER_PROOF</code></td>
</tr>
</tbody>
</table>
MODEL OF THE SECRET CHANNEL

1. Child center sends a ticket-granting ticket (TGT) to TGS.
2. After verifying that the TGT is valid and that the child center is permitted to access trusted center, the TGS issues ticket and session key to the client encrypted with child center key and TC key accordingly.
3. Child center sends the ticket containing session key and a time stamped service request, encrypted with this key, to the TC.
4. TC decrypts the ticket with its own secret key to retrieve the session key, decrypts and checks the service request with it. If the request is correct, it sends the confirmation to child center encrypted by session key.
Distant Modeling of NIZK Protocol for Public Keys Certification

ABOUT SOUNDNESS OF NIZK Protocol for Public Keys Certification

\[ \delta = ((2^t - t - 1)(2^t - t)\ldots 2^t)^{-1} << 2^t. \]
CONCLUSION

The distant educational model of a relatively complex cryptographic protocol is represented.
It can be used only for education because transmissions are public knowledge and can be observed by all students.
Moreover, some secure parameters generated artificially inside the protocol whereas actually they should be created by a higher center.
Nevertheless, it allows us to show many cryptographic primitives and to master practically such cryptographic constructions as non-interactive zero knowledge proof and the designated verifier proof.
It was shown that the soundness of the protocol considered in this paper is much less than the soundness of the original Schnorr protocol.
Thank you for attention!

Speaker’s contacts:

Alexander Vinnikov,
SoftEngineering Ltd
al.vin@bk.ru
soft-e.ru