AIMMS User's Guide - Project Security

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

Chapter 19

Project Security

Project security

When you are creating a model-based end-user application there are a number of security aspects that play an important role.

- How can you protect the proprietary knowledge used in your model?
- How can you prevent the end-users of your application from modifying the project (thereby creating a potential maintenance nightmare)?
- How can you distinguish between the various end-users and their level of authorization within your application?

This chapter

AIMMS offers several security-related features that address the security issues listed above. These features allow you to

- encrypt the source code of your model,
- introduce authorization levels into your model, and
- set up an authentication environment for your application.

This chapter describes these mechanisms in full detail, together with the steps that are necessary to introduce them into your application.

19.1 Encryption

Encryption . . .

If you want to protect your investment in model development, the easiest way to accomplish this protection is to use the encryption scheme discussed in this section. Note that project access to the project and model is unconditionally prohibited in an encrypted project, even by the developer of the model himself.

Several ways of encryption

AIMMS supports several manners of encryption of project and model source files, including your model source. Please note that AIMMS will only encrypt `.aimms`, `.libprj`, and `.ams` files. All other files that are exported (including user files that reside in your project file) are not encrypted. It is up to you to choose the encryption scheme that works for you.

- **Standard encryption:** results in an end-user version of your application that can be run by everybody.
- **Password protected encryption:** results in an end-user version of your application that can be run by anyone who knows the password. Upon starting of the application the user is prompted for the password.

- **Key-based encryption:** result is an end-user version of your application that can only be run by users whose public key was present in the key folder that was specified during encryption. The users need to store their private key in the `ApplicationKeys` folder on their local system or, in case a license server is being used, on the system on which the license server is running.

Exporting your project

To ship your application for end-user deployment you should export your application as a single `.aimmspack` file (see also Section 15.2). By combining the export with one of the available encryption schemes you simply produce an ready-to-ship version of your application in which the source of your project and model files is securely protected.

Export ...

You can create such a single `.aimmspack` file version of your application through the **File-Export End User Project** menu, which will open a **Select Destination .aimmspack file** dialog box. This dialog box requires you to specify the location and name for the `.aimmspack` file.

... and encrypt

Having specified the name for the `.aimmspack` file, the **Encryption of Exported End-User Project** dialog box (as illustrated in Figure 19.1) opens and allows you to add encryption to the exported version of your application. Select one of the available encryption schemes and specify all relevant missing information (e.g. passwords, a folder containing the public keys of your users).

Restrict access to a specific license number

In addition to encrypting your application, you can restrict access to your application such that only users whose AIMMS' license number lies within a specified range can run the application. This prevent the application from being run by other AIMMS users, even in case a password or private key has been compromised.

Add an expiration date

If you add an expiration date to the encrypted application, AIMMS will not allow your end-user to run the application after that specific date. In addition, you can have AIMMS warn your end-user about the expiration date if the application is started within a specified number of days of the expiration date.

19.1.1 Public key encryption

Public vs. private keys

AIMMS' key encryption uses a common public key algorithm which assumes the presence of two associated keys, a *public key* and a *private key*. Anyone who has access to a certain public key can encrypt data, but only the owner of the corresponding private key can decrypt the data. So, if you want someone to send you encrypted data, you should share your public key. At all means, a private key should be kept private.

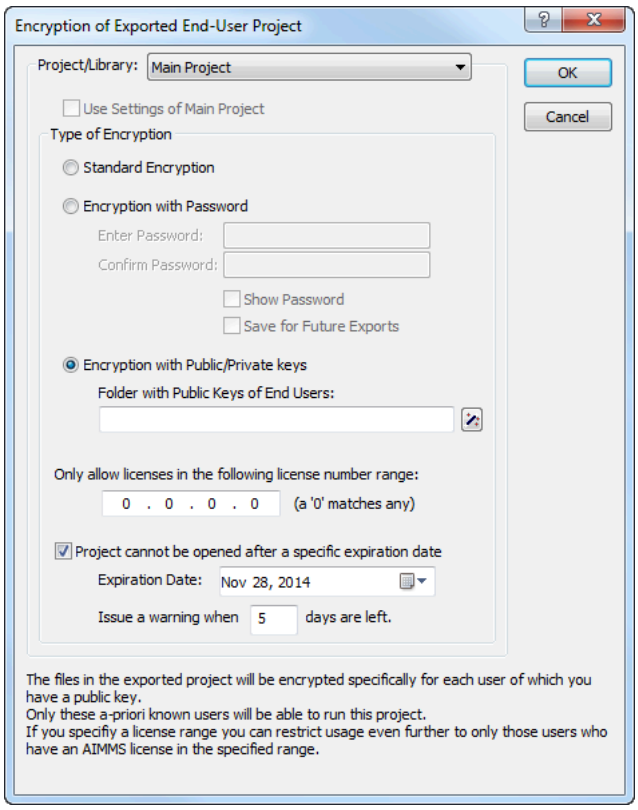

Figure 19.1: The **Encryption of Exported End-User Project** dialog box

Creating a key pair

Through the **Tools-License-Generate Public/Private Key Pair** menu, you can generate two associated key files.

Encryption using multiple public keys

An application can be encrypted using a collection of public keys. The resulted encrypted application can then only be run by any private key, matching one of the public keys in the collection that was used during decryption.

Private key folder

When attempting to decrypt an application, AIMMS will look for matching private keys in the `AIMMS\ApplicationKeys` folder. The folder is located as a subfolder of the folder described by the `ProgramData` Windows environment variable. On a typical Windows 7 or Windows 8 system, this private key folder is `C:\ProgramData\AIMMS\ApplicationKeys`. In case a license is provided over the network by an AIMMS network license server, the private key to decrypt the application may also be present on the system that runs the license server. In case the private key is provided by the license server, only users that are granted access to a network license on the server, may use the private key from the server.

19.1.2 Encrypting your application: some use cases

Use an existing public key

To encrypt an application for a specific user that has already created his own key pair, just request the user for a copy of his public key and use the public key to encrypt your application.

Use as application license

In case you generate a new public/private key pair yourself and use the newly generated public key to encrypt your application, the corresponding private key serves as an *application license*: As soon as you provide an AIMMS user with this private key (and access to the encrypted version of your application), he will be able to run the application. In this scenario, it is even possible to generate a collection of key pairs in advance and distribute a new *application license* anytime you get a new user for your application.

Use in an AIMMS PRO environment

When publishing an application on a AIMMS PRO server, you are advised to encrypt your application using the public key of the AIMMS network license server that is used in the PRO configuration. After that, any user who has been granted access to the PRO server (and the specific application), is able to run the the encrypted application, without the need to have a public/private key pair of his own.

19.2 User authentication and authorization

User authentication

When an application is set up for use by multiple users through AIMMS PRO, it is usually considered desirable to limit access to the application to particular (groups of) users, make sure that users have access to only those parts of the application that are of interest to them, and can be given or denied the right of access to each others data.

Authorization via AIMMS PRO portal

When publishing an application on a AIMMS PRO server, you can manage access to your application through the AIMMS PRO portal. The AIMMS PRO User's Guide http://manual.aimms.com/pro/ decribes more details about the setup of users and groups for your application.

Authorization via model

Next to arranging access to your application application-wide through the PRO portal, the PRO system library extends your model with functionality to access user- and group-related information from within your AIMMS application. More, specifically, through the PRO library function

```
pro::GetCurrentUserInfo
```

you can retrieve the currently connected PRO user name and the PRO group membership of the currently connected user.

Role-based security

Using the PRO user name and groups discussed above, you can set up your own customized role-based security scheme within your application. You can accomplish this by associating roles within your application with group membership of particular groups defined through the user management facilities in the AIMMS PRO portal. If PRO user management is linked to your Active Directory environmoment, role-based authorization to your application can also be arranged directly through your company's Active Directory environment.

Example

Assume that ExecutionAllowed is a two-dimensional parameter defined over a set AllApplicationRoles declared in your model, of which the actual set of PRO groups, retrieved via pro::GetCurrentUserInfo, is a subset, and a user-defined set of application-specific ActionTypes. Then the following code illustrates the to allow or forbid a certain statement to be executed in a role-based manner.

```
if ( exists(appRole | ExecutionAllowed(appRole, 'Solve') ) then
    solve OptimizationModel;
else
    DialogError( "None of your application roles does allow you\n" +
                 "to solve the optimization model" );
endif;
```

Use in the interface

You can also use parameters defined over AllApplicationRoles to influence the appearance and behavior of the end-user interface. More specifically, the following aspects of an AIMMS end-user interface can be influenced through the nonzero status of (indexed) parameters:

- the access to a page through the page tree-based navigational controls,
- the visibility of graphical (data) objects on a page,
- the read-only status of data in a data object, and
- the visibility and enabled/disabled status of menu items and buttons.

Note that both the Windows and browser-based AIMMS UIs support such dynamic model-based access controls.