Secure Exchange of Black-Box Simulation Models using Functional Mockup Interface in the Industrial Context

Christian Wolf[1] Miriam Schleipen[1] Georg Frey [2,3]

[1]EKS InTec GmbH, Germany, {christian.wolf,miriam.schleipen}@eks-intec.de
[2]Chair of Automation and Energy Systems, Saarland University, Germany, georg.frey@aut.uni-saarland.de
[3]Department of Industrial Security, Center for Mechatronics and Automation Technologies (ZeMA), Germany, georg.frey@zema.de

Abstract

Functional Mockup Interface (FMI) is a standard for exchanging simulation models described as Funcional Mockup Units (FMUs) in a platform-agnostic way. FMUs can be implemented as white-box or black-box models. In the industrial context, it is common to exchange black-box models between partners to hide intellectual property. Using and running such models, though, is a security issue as there is no way to verify and validate the content of the models. This security issue must be addressed especially in the industrial context where security is considered high priority in general. Based on an exemplary model exchange, possible attacks and possible countermeasures are analyzed in this work. By using cryptography, three different approaches to pack the additional metadata are presented that aim at providing end-to-end integrity checks to a black-box simulation model. Together with administrative measures, this allows to define those FMUs to be trusted and executed. For the sake of completeness, a prototype was implemented to help with the cryptographic processes and show the effectiveness of the provided solution.

Keywords: Simulation, Security, FMI/FMU, AutomationML, Black-Box Model, Certificate, PKI

1 Introduction

In the current design and development process of products and goods, simulation provides a way to verify functionality and optimize in an early stage of development. For the sake of reducing the involved manpower to build such simulation models, these models are typically shared between stakeholders. Current developments like the FMI standard show the need to provide a standardized way to simplify model exchange. However, not all models can and should be provided as a white-box model, e.g. there might be restrictions on who should be able to read and understand the model in depth.

Current practice is that FMUs are transmitted via email or download from the vendor website. Although there might be SSL-based security added on the point-to-point transports, like HTTPS, these processes must be considered insecure as there is no end-to-end guarantee of the security. The FMU stored on the web server could be changed intentionally or by accident without notice.

One way to tackle the problem would be to use secure data spaces where the platform itself is designed for security, see Christoph Schlueter Langdon and Karsten Schweichhart (2022). On such a platform, access is coupled with the trust in the corresponding entity, so within this data space everyone can be trusted. One project to create a safe space is Catena-X (Hedda Massoth 2022).

By closing down the models in form of black-box simulations, it is (ideally) impossible to check the model for correctness, feasibility, and security. There might be arbitrary code included in the model that will get executed once the simulation is run. There are approaches to reduce the risk by e.g. running the model in an secured, immutable environment ("sandboxing" the model) but this is prone to bugs and should only serve as a last resort. Instead, this paper presents a method that allows to check statically the validity of a simulation model before even running it.

In this work, some basic tools and concepts from various domains are presented in Section 2. The underlying problem of this paper is presented in Section 3 and some risk analysis on possible attacks in the state of the art carried out in Section 4. To tackle the issues, some abstract considerations in Section 5 as well as some rejected considerations in Section 6 are used to derive multiple different propositions in Section 7. Finally, a proof-of-concept prototype is presented in Section 8 and an outlooks as well as a summary of the results is given.

2 Preliminaries

Throughout this paper, concepts from different domains need to be combined. In this section, the basics of different fields are covered for later connection in Sections 5 through 7. By integrating cryptogrphical methods with simulation models/FMUs, new benefits can be generated.

2.1 Functional Mockup Units

Functional Mockup Interface (FMI) is a standardized way to exchange simulation models for various purposes. The current version 3 is described in *FMI V3* (2023). A single model is called a Functional Mockup Unit (FMU).

An FMU is technically speaking a compressed folder with some files inside. There is for one the manifest, an XML file describing the model. In the manifest the inputs and outputs are specified, as well as parameters and other options.

The actual implementation of the model logic can be provided by means of embedded source code or a precompiled library (DLL or shared object depending on the operating system). It is also possible to have both or a mixture of these approaches.

There are in fact three types of FMUs available: model exchange, co-simulation, and scheduled execution. Each type has its dedicated use case. While the co-simulation type brings its own mathematical kernel and is run as a dedicated process during execution, the other two types run within the process context of the host's simulation environment.

There exists the *FMU Trust Center* presented by Johannes Mezger et al. (2011). This allows to encapsulate the FMU under consideration into its own trusted environment. By using a dedicated and secured infrastructure, the FMUs can be encrypted during transport. Only during the time of simulation, a decrypted version is existing. That way, one can provide encrypted FMUs without the risk of sharing internal knowledge.

2.2 Digital Twin and Meta Models

In typical industrial applications, there is the need for a common description of arbitrary data in a structured way. This allows to exchange data in generic ways and is commonly referenced as a *digital twin* (DT).

Currently, two major meta models to describe such digital twins are established: AutomationML (AML, Rainer Drath (2021)) and the Asset Adminstration Shell (AAS, Deutsches Institut für Normung (2019) and *IEC 63278-1* (2022)). Both models have a similar goal and are interchangeable in terms of this paper. Thus, throughout this publication only AML is going to be used as an example. Similar results can be obtained with AAS with slight modifications.

Coming from the DT context, these meta models provide a method to describe things in a portable way. By defining common file formats the problem is reduced to understanding the semantic of the various parts. Instead of defining their own vocabulary, these models rely on existing descriptions. For example, simulation models can be integrated as FMUs.

AML uses XML files and all linked files just are connected using so-called *external interfaces*. There is however the option to embed all relevant files into a bundle called an *AutomationML container* (Rainer Drath 2021). Such an AMLX file is a zipped folder with all attachments plus additional files to satisfy formal requirements. According to Rainer Drath, Markus Rentschler, and Michael Hoffmeister (2019), both AMLX and AASX (from AAS) files comply with the Open Packaging Conventions (OPC) in accordance to *IEC 29500* (2012).

2.3 Security

For computing systems, there exist various requirements that a typical user assumes but that need careful planning and engineering. In Avizienis et al. (2004) these are categorized into dependable and secure aspects.

Dependable in this context means that a system is behaving as expected and not causing any major risks during operation. This is mostly the same definition that we would call a physical product *functional*: a freezer must keep the goods frozen even during e.g. an outage of a half an hour while not emitting any toxic gases to the environment.

There however is the concept of security that mainly became common with digitalization. Security breaks down to three main aspects: confidentiality (Is my data safe and nobody can access it?), integrity (Can I trust my data or has anyone tampered with it?), and availability (Can I access my data anytime I want?). Typically, not all of these goals can be achieved 100 % at the same time.

Some methods to establish these aspects of security are hashing, asymmetric cryptography, and certificates. The following sections will give a brief introduction for the reader.

2.3.1 Hashing

A hash function is a cryptographic algorithm that can be applied to data and files and that results in a deterministic value, aka there must be no randomization involved (Bart Preneel 1994). There are different hashing functions commonly used also with different goals and motivations involved like SHA-1, SHA-256, BLAKE-256, etc (Jean-Philippe Aumasson et al. 2008; Alex Biryukov, Dumitru-Daniel Dinu, and Dmitry Khovratovich 2015; Rajeev Sobti and G Geetha 2012; Stefan Tillich et al. 2009).

The output of such a hasing function is called a hash and of fixed size. For example, the SHA-256 hash function uses 256 bit to represent a hash. This is true for arbitrary long input data sequences, so the function cannot be injective: multiple input streams can result in the same hash.

This property of non-injectiveness has multiple effects. First, hashing cannot be undone. Once a data steam has been hashed it is no longer possible to conclude the original data. Hashing is thus a one-way function.

In order to compare the content of files/data streams, comparing the hashes is one typical method. One has to keep in mind that comparing function values and deduce the equality of the function arguments only works for injectve function by definition. For non-injective functions, like the hashing functions, there is a (small) probability that two distinct data streams result in the same hash, called a *hash collision*. Therefore, hash functions are constructed such that small changes in the input data (even as small as a single bit) lead to significant changes in the resulting hash value. This should reduce the probability of small defects to get unnoticed.

DOI
10.3384/ecp204487

One typical use case for hashing is to establish integrity of data. Unless there is a collision, one can identify and reject a compromised data stream if the hash is known.

2.3.2 Asymmetric Cryptography and Keys

The RSA algorithm named after Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman (1978) is a well-known method for asymmetric encryption and signatures. In the meantime there were different variants developed to overcome some shortcomings in the original version. These modified versions serve as a backbone of the current internet encryption in e.g. HTTPS, IMAPS, or other encrypted protocols.

All cryptographical methods base on pure numbers, thus all data is interpreted as such (big) numbers. There are now two different operators that are typically used in cryptography: encrypt/decrypt and sign/validate.

Assume a (big) number M that contains a message. By using a publicly known mapping $E(\cdot)$, anyone can *encrypt* M and will obtain $M_{\text{enc}} = E(M)$. The mapping however is designed to not be reversible in a simple manner. By only knowing M_{enc} one cannot deduce the value of M. The intended recipient of the message knows another mapping $D(\cdot)$ that allows to *decrypt* the message again $D(M_{\text{enc}}) = D(E(M)) = M$. Of course, the mapping $D(\cdot)$ must be kept secret.

The RSA algorithm provides an implementation of this scheme by using big prime numbers and some properties of integer arithmetic. The important part is that the message M_{enc} can be transmitted over insecure lines and its confidentiality is guaranteed by cryptography.

When the mappings $E(\cdot)$ and $D(\cdot)$ are constructed accordingly, it is possible that they are mutual inverses

$$D(E(M)) = M = E(D(M)).$$

That means that applying both operations once will cancel each other independently of the order of application. If the decryption is applied on M, this is called *signing* and the later encryption process *verification*.

The signing can only be carried out using the private mapping $D(\cdot)$. This allows to create a proof that a certain message was written by a certain person. Everyone can verify using $E(\cdot)$ if the message was changed. Thus, this method allows to ensure integrity using cryptography.

In the RSA algorithm, the mappings E and D use specially crafted numbers. These numbers are called public (in case of E) and private (in case of D) keys. These keys can be seen as parameters to general functions to obtain the individual implementations of the two mappings E and D mentioned above.

If the private mapping $D(\cdot)$ gets compromised (leakage of the private key) or the encryption method itself gets broken, the methods do no longer work. In case of encryption, the unencrypted message might be leaked. in case of signatures, an attacker might fake valid signatures.

For the sake of speed and memory consumption, these methods are typically combined with other cryoptographic methods that use symmetric cryptography. These details are more implementation-related and do not play a significant role for the whole process when implemented accordingly. They will therefore be neglected here.

2.3.3 Certificates and PKI

In order to simplify the management of public keys to be trusted, a new infrastructure needed to be built. A first step is to encapsulate the public keys into another structure to attach some meta data with them. Such a structure is called a X.509 certificate (Sharon Boeyen et al. 2008). The infrastructure to manage and use these certificates is called a *Public Key Infrastructure* (PKI).

Each certificate contains (apart from the public key) some additional information: there are some pieces of technical data (like the key size) stored. Also, there are a few human-readable strings describing the person or organization the certificate is associated with. These attributes are defined in Sharon Boeyen et al. (2008) and contain e.g. the *country*, the *common name*, and others. There are standardized abbreviation (`CN` for common name, etc). The attributes are rather restricted in terms of size and type: only 64 byte of ASCII text are possible per attribute for simple fragments of information like mail addresses or host names.

Additionally, certificates are issued by some authority. Each certificate has exactly one issuer. Issuing a new certificate is twofold: First, all relevant data like public key, meta data, etc is combined in a so-called *certificate signing request* (CSR). Next, the issuer uses his private key to sign the CSR after checking it. The new certificate is built from the CSR, the signature, and some unique reference to the issuer.

This principle of issuing certificates is its strength: it allows to delegate trust from a single source (so called *trust anchor*) recursively. Any user can trust some authorities, typically done by the operating system. Every service just needs to provide a chain of certificates to such a trust anchor, a so called trust chain. In that way the user can trust a public key without manually verifying the authenticity of said public key.

One last link is missing in order to establish trust in a thing, a connection, a name, or anything outside the cryptographic environment. The trust so far is *in a certificate*. For example, to obtain a certificate for a dedicated host name, the CN of the underlying CSR must have been set to the host name. This connects the certificate with the host name: a HTTPS connection requires that the `CN` of the signed certificate is equal to the host name. There are extensions for more complex scenarios that allow multiple host names per certificate (see `subjectAltName` in Sharon Boeyen et al. (2008)) but this is out of scope for this paper.

2.4 Intellectual Property

In Keith Eugene Maskus (2000), intellectual property is defined as following: “Human thought is astonishingly

creative in finding solutions to applied technical and scientific problems [...]. These intellectual efforts create new technologies [...], develop new products and services [...]. They result in intellectual assets [...], that may have economic value if put into use in the marketplace. Such assets are called intellectual property to the extent they bear recognized ownership."

All pieces of software, tools and knowledge obtained by a company in its productive effort is considered to be owned by said company. The company will use these pieces for further benefit and protect them against access by third parties. This is intellectual property (IP).

3 Problem Formulation

Let's assume there are two (industry) partners A and B. One of them (A) is a vendor that sells some products. The other one (B) is a big player that buys these and integrates them together with dozens other products into complex systems. In order to simplify and optimize the engineering process, A provides simulation models for the relevant products to B. In the rest of the paper, we are focusing on one such model.

3.1 Concerns of Model Provider

These models are generated by the staff of A and contain parts of the internal knowledge of A. There could e.g. be some fancy algorithm implemented in the controller firmware of these products. The simulation model will eventually mirror these controller features to provide a good match of the model with the real world. Thus, part of the simulation model would probably be the proprietary firmware which A considers IP. As a result, partner A might have concerns to publish a white-box model to the partner B. Speaking in terms of security from Section 2.3, the partner A has a high requirement for confidentiality.

There might be additional wishes by the model provider A. For example, a model should have a lifetime after that it should be updated to the newest version. There might also be companies whose business model is to sell model hours.

3.2 Concerns of Model User

After transmitting the model to partner B over a to-be-defined transport, there are also some other aspects to be taken into account. At first, the model must be running in general and according to the specification. That is mostly covered by the dependability. However, there are also security considerations: The model might, in general, contain malicious code that is executed on the infrastructure of partner B. These threads range from attaching spamware by accident to active attacks (e.g. distribution of ransomware by a former employee).

So, from the perspective of B there are other requirements.

- Integrity: The model in question should be exactly the model of the product and not be tampered with.
- Availability: The model should run anytime the partner B needs to run simulations without interfering with or blocking B's infrastructure.
- Confidentiality: The complete simulation model, where the model of A is integrated to, might be IP-related from the perspective of B. No data should be extracted using the simulation model.

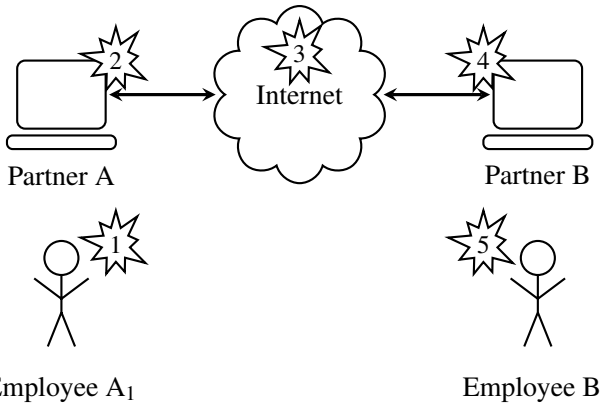

Figure 1. Data flow of the simulation model with possible attack vectors.

3.3 General considerations

Having seen the positions of both parties one has to realize there are multiple levels of concerns. On the administrative level the partners use legal contracts to fix their mutual responsibility and accountability. This results in (internal) policies about what model from which partner might be run in which context. The technical level is located below that to ensure the correct realization of the administrative decisions and policies. This paper focuses on the technical level on how the parties can establish trust in the exchanged model. Therefore, it is assumed that both parties can be trusted and an appropriate policy is in place.

In that sense the exchange is very similar to sending potentially malicious data like documents with macros enabled, DLLs/SOs, etc via email. In contrast to best practices (e.g. only opening documents with macros globally disabled), exchanging FMUs in this way will enforce the user to actively open such untrusted documents.

4 Attack vectors

There are multiple parties involved, thus the analysis of possible attack vectors need to be carried out for all of them.

Starting with the process description in Section 3, one might follow the path of the simulation model. In Figure 1 the data exchange between the participants is depicted as well as the possible points of attacks. The possible attacks are numbered to simplify referencing them later.

1. The first opportunity for an attack is the user A_1 who generates the FMU on the side of A. He might will-

DOI
10.3384/ecp204487

ingly or accidentially add malicious code to the FMU to be shipped.

2. The next opportunity is the infrastructure of company A. Although A_1 does his best in order to ship the FMU in a valid state, e.g. a virus on his machine might leak into the process and thus compromise the generated FMUs.

3. Knowing the dangers of the current time, the internet cannot be considered a safe place. Traffic might be transmitted over insecure channels that are not end-to-end encrypted. This also covers simpler methods like spoofing messages or just crawling and collecting data.

4. Analogously to attack 2, the infrastructure of B could be affected.

5. Finally, the user B_1 might be responsible for the potential attack similar to vector 1.

These attack vectors are not exhaustive, in fact there are attacks possible that combine different approaches. An example of such a combined attack would be the following: via social engineering the attacker could gain knowledge on A_1 (attack vector 1) to get known about a common project. Using a forged mail (attack vector 3) he tricks B_1 into opening and starting the model (attack vector 5).

For almost all possible attack vectors, there are concerns for both parties A and B involved. Some cases do not make sense, though: exposing some IP of A by A_1 can be done in various forms and does not fit in the context of this paper. Similarly, the IT department of A will do its best to prevent attack vector 2 to be used against A. The same is true for vectors 4 and 5 with respect to protection of B.

5 Abstract Solution Approaches

There are some general approaches that should be considered before the implementation can be done.

First, in order to simplify implementation effort on both sides A and B, the FMI standard is used as a basis. This directly addresses the requirements of the partner A with respect to confidentiality: the FMU can contain precompiled versions of the models leading almost to black-box models. In theory, one could reverse-engineer the binary codes and obtain knowledge. There however are options to obfuscate and encrypt binary code until used (Michael Klooß, Anja Lehmann, and Andy Rupp 2019; Thomas Agrikola 2021). In order to use such methods, it is required for the FMI standard to support such enhancements officially: the official standard (*FMI V3* 2023) allows to augment functionality by using so called layered standards, the resulting unit must, however, still be compliant to the basic standard. If the encapsulated binary code was encrypted, this is no longer covered by the standard. The simulation environment needs to be able to decrypt accordingly on the fly in order to make any use of the FMU.

The primary goal for B is to establish trust on technical level between the partners A and B. This should be done using cryptography: a method is to be derived that can crypographically prove the integrity of the model and its origin (partner A). No model should be run that was tampered with or that was broken during transport.

The most prominent issues come from attack vector 3, potentially in combination with other ones. Apart from that, it is typically sufficient to require trust between A and B: by contracts and legal bindings these parties typically consent on mutual trust. The trust of B in A_1 is not needed as A will trust A_1 and hold him liable and responsible for his actions. Also, from B's perspective, still A is liable for any issues. The same holds true for the trust of A in B_1. Therefore, in this work the attack vector 3 is to be considered the primary one.

In order to prevent the model from accitential changes, a hash of the model is generated directly after forging the FMU by A. This hash can be delivered with the model and checked just before the real execution of the model. The model verification will detect changes to the model with a very high probability (i.e. if there are no hash collisions) and report that. It might be up to said user how to cope with such a situation but this decision is part of the administrative policies excluded from this paper. In fact, the tests are just an addon to FMI and can always be overridden by the simulation environment.

So, in general the hash allows to detect transmission and storage errors between A and B. In case of a malicious actor, the hash will only provide little help: the attacker could simply calculate the hash of the modified FMU and replace the original hash as well. The user will check the (modified) FMU and compare with the faked hash. Thus, no warning will be issued and the user might be running the tampered FMU without further notice.

To prevent such an attack, it is necessary to establish trust in the hash. This can be achieved by crypographical means in the form of certificates. The vendor A creates a valid certificate (plus its private key) before deploying the FMU. The private key is used to sign the hash. This certificate with the complete trust chain including the signature of the hash is delivered with the FMU.

On the user's side (B), first, the hash is validated. To do that, the hash of the FMU is calculated by B. If the signature matches with the calculated hash, B has proven that the person who created the signature had access to the corresponding private key.

It is not required to provide a hash of the FMU in the metadata as long as there is a croptographic signature available. However, deploying the hash as well has the benefit that the test of the validity can provide more detailed error messages on why the validation failed. With the hash it is possible to distinguish an accidential trans-

mission error from an attack that might need further investigation.

This process of authentication is however not sufficient to provide authorization to run the FMU. Here, the administrative policies as depicted in the problem formulation need to be addressed. The question is: which entities are to be allowed to provide trusted FMUs and how to prevent other entities from pretending and faking their identity? Typically, this is done by enforcing certain rules on the certificate chain. For example, there could be a requirement that the certificate is (indirectly) signed by a certificate with a well-known CN. If the systems are online, one can use authentication using other systems, but this is not the major point in this paper (see e.g. challenges `HTTP-01` and `DNS-01` in Daniel McCarney (2017)).

In any case, the simulation environment on the infrastructure of B must run these tests just before the FMU gets loaded and started. This is similar to the signing process of system libraries in the Microsoft Windows operating system that refuses to install unsigned libraries. These checks are in fact a requirement for the software providers that handle the FMUs. These providers need to add support for appropriate steps in order to ensure security of the models, potentially establishing an accompanying standard to the plain FMI standard.

It is vital to understand that once the simulation environment separates the simulation model from the security data by checking the signatures (and only passing the simulation model on), the protection ends. For example, assume that party B checked some FMU by whatever means, decides the FMU to be secure, and stores the FMU in a local simulation library without the security data. The attack vector 1 through 3 has been ruled out so far. However, the attack vector 4 is still open: changing the FMU in the library could easily get unnoticed. Thus, the separation should be done as late as possible to cover most of the possible attack surface. Ideally, the check is done right before the simulation itself is run and the first call to a function in the FMU is carried out.

Note, that the suggested approach does only secure the transport of the models. The execution is not affected, which can be seen that the existing simulation cores do not need to be altered. So, there is no protection against online-changes to the FMU while it is running, e.g. due to defective RAM.

The approach of using certificates also allows for the additional feature of lifetimes of the models. All certificates have a lifetime which ends at some time. This can be used to invalidate a model after a certain point in the future. The checker in the software will refuse the validity of the certificate due to an expired certificate if the lifetime of the model has expired. This is however not cryptographically enforced.

There is no guarantee about confidentiality so far from the perspective of A. Anyone who has access to the FMU will be able to run the system. To prevent that, one could use encryption on top of the provided solution. The encryption is not as simple as signatures as each legitimate user of the FMU needs a way to decrypt it with their individual private keys. One possible solution was that vendor A provides a public API where any potential customer (like e.g. B) can request an individually encrypted simulation model. To do so, the potential buyer needs to authenticate. Before the model is encrypted and provided, the vendor can check if the request is authorized.

The certificate chain can be adopted to the needs of the use case. If the vendor A provides individual certificates for their employees, A_1 would use his personal certificate (and key) to sign the FMU certificate. This means an additional benefit of accountability: whenever a problem arises it can be tracked down to the individual user A_1 who has signed the FMU in question. This simplifies incident analysis and provides some internal mutual protection between A and A_1.

By defining an extension to the FMI standard, one could introduce a formal certification scheme: FMUs could be labeled as “certified secure according to the standard”. This could also be applied for the simulation tools to certificate that these abide some security guidelines about when a model is considered harmful and not to be run.

6 Alternatives considered

There are a few alternatives to the abstract solutions presented above. These can be ruled out for different reasons and the argumentation should be mentioned here, shortly.

6.1 Classical transport with cryptography

The most direct approach to this problem would be to use state of the art mechanisms provided by the IT to establish cryptographal security. One could use S/MIME (Blake C. Ramsdell 1999) or GPG/PGP (Simson Garfinkel 1995) to add end-to-end encryption to e.g. mail delivery. This has however the drawback that the signatures are typically bound to a single person instead of the corresponding company. Volatility in personnel will make handling hard and error-prone.

Additionally, these approaches need manual work by the users. While this seems only a minor burden, it adds the risk of wrong application and user errors. In case of problems, people might tend to avoid the system altogether. A fully automatic solution is preferable here.

6.2 Callback in FMU

Looking at the problem from the FMI context, one straight forward approach would be to put the security information into the code. The user's simulation environment could use `fmi3GetBinary` to extract and check it. This will however not serve well as the function call would already execute code from the FMU that is (not yet) to be trusted. So, in order to check the validity, the simulation environment must only use statically available information.

After the security has been established by other means, there are still use cases to call a custom callback in the FMU. This would allow to realize the already mentioned

DOI
10.3384/ecp204487

business models that require the validation of a license of the customer. So, after the initial security check, the FMU itself could issue a check if the user (B) is to be allowed to run the simulation and abort if not. There could be some license fees to be paid or simply the model must only be run on behalf of real customers. The checks could be arbitrary to match with the business model of A. Currently, different other callbacks are (mis-)used for such functionality, having a dedicated callback would be preferable.

6.3 Storage of hash in CN

One has to note that the attributes of a certificate are generally not potent enough to store the complete FMU. This is true for the proposed solution as well.

Complying with the de-facto standard of storing in the CN a unique identifier, one could try to calculate an immutable identifier for an FMU using hashing. Then, this hash could be used in the CSR to generate a certificate with the CN set to the hash of the underlying FMU. That way, the certificate would not just be a generic certificate but the certificate of said FMU. In that sense, it would avoid to use external signatures and keep all cryptography in the PKI.

By looking at the recent hash functions SHA-512 or BLAKE-512, these use 512 bit or 64 byte in binary form to represent such a hash. As the attributes are only ASCII text, the binary must be mapped into ASCII which enlarges it further. For example, to represent the complete hash one could use base64 encoding but will then need 88 characters (86 as the length can be considered known). Considering that the attributes are capped at 64 characters, the hash will not fit. Consequently, one could only use hashes with 384 bit or less.

Larger hashes provide better security in theory and make (exploitable) hash collisions less likely. The size of the hashes was therefore growing in the past and one has to assume that this trend will continue. So, one must at least consider larger hashes and cannot reject them in the architecture of this approach.

7 Proposed Solutions

The so far described approaches are of rather abstract nature. There are different ways thinkable how this could be implemented and especially where the hash and the signature could be stored.

7.1 Adaption in a layered FMI standard

The most simple way would be to implement the hashing and signing inside an extended layered FMI standard. That way, the FMU would still be an encapsulated and complete model that has the security features included. The FMI standard allows to provide additional data like static (XML) files in an appropriate folder under the top level folder called `meta`. For example, one could define a folder `meta/de.eks-intec.fmi-sec` that contains further files with the required security-related data.

The trivial approach of hashing the complete FMU has one major drawback, though. In order to zip the FMU file, one needs the hash, leading to an chicken-and-egg problem.

Consequently, one needs a more fine-tuned approach. Just before packing of the FMU, all included files are listed. The algorithm will filter out the files in the folder `meta/de.eks-intec.fmi-sec`. For each file, a hash is calculated and is stored individually with the corresponding (relative) filename in the `hashes.xml` file in said folder. Having finalized this file, one can calculate a hash and a signature of `hashes.xml` and write these into a sibling file `security.xml`. The latter is augmented by the complete certificate chain to help validating.

To check the validity of such a composed FMU, the simulation environment has to carry out multiple steps: first, the validity of the certificates need to be ensured (like checking for expiry, trust, and pairwise signatures). Then, the certificate can be used to check the validity of the `hashes.xml` file. Once this is confirmed, each file in the FMU has to be checked against the stored hashes. There should be no additional files found, so, if that happened, the algorithm would issue a warning or even fail validation of the complete FMU. Once all files in the FMU have been validated, it can be considered harmless and processed further. In contrast to the approach in Section 7.3, this one does not need additional tools (e.g. to read AML files).

7.2 Externally in the Network

Another option to handle the hashes, certificates, and signatures would be to define one URL per FMU that will provide all relevant information. So, the software running the model in question would need to download the security information from the site and check the FMU against that.

This is possible because the URL can be pre-defined and thus statically embedded in the FMU somehow. This might work similar to the approach in Section 7.1. In the open source world it is common to have for each downloadable file an additional file with some hash or signature to check for download errors. In a similar fashion, one could manage the deployment of the security-related data of FMUs.

As the security information is generally available online, this process allows also to update the certificates (renew it in case its lifetime should be extended past the original end). Regular updates of short-lived certificate enhance the overall security (Emin Topalovic et al. 2012; Ronald L. Rivest 1998): the longer the certificates are valid, the more time is to break the keys. Short-living certificates and keys reset these time windows and make breaking the keys by pure chance very unlikely.

As the vendor A has control over the certificates, it is also possible to think of new business models. One could pay per issued certificate, per hour of model usage, per simulation run, or other metrics. The need to fetch a certificate makes it rather simple to control these type of

usage-based billing. The online approach works similar to a physical token but without the overhead of physically transferring it.

The drawback of this method is that the simulation environment needs to have continuous access to the certification service. This makes it impossible to use air-gapped systems. There could be caches included that provide a way to store the certificate until it expires, still, the renewal process needs to be triggered sometime. Also the certification provider (typically A) could anytime stop issuing new certificates (e.g. also due to technical issues or bankruptcy) and the certificates as well as the simulation models will cease to work.

7.3 Embedding in a Digital Twin

By combining technologies of different fields, one might achieve a matching solution for most of the problems presented. Instead of embedding the data in the FMU as described in Section 7.1, there might be already a location present to store these additional information. According to Roberto Minerva, Gyu Myoung Lee, and Noel Crespi (2020) many products are in the meantime supported by their individual DT. Using the example of AML as a description language for DTs, one has to realize that for many use cases (not only in the context of this paper but for general problems), on might fall back to pre-existing solutions.

In AML, e.g. a library to import FMUs into the AML file exists already (Olaf Graeser et al. 2011). If the vendor A already provides a DT for his products, the FMU can easily be added into said DT. If no DTs are yet existing, the overhead to create such a twin is minimal.

With the AML in place, one has a clear and well-defined outer shell. The security elements in AML need to be modeled separately, the basic elements are certificate chain links. One example modelling of such a DT can be seen in Christian Wolf, Miriam Schleipen, and Georg Frey (2023).

The benefit to use a dedicated format over extending the FMU in 7.1 is that other modelling standards (like e.g. SSP (*SSP* 2022)) can be used without change. This makes this approach a very generic one.

For the AAS, Andre Bröring et al. (2022) present an approach that allows to prove integrity of the data. However, the basic idea is to mimic the GIT version control system in terms of a DT meta model. As with GIT, it is possible to rewrite the history to insert a tampered FMU unnoticed. The integrity of the history and trust in the FMU can thus only be guaranteed for read-only access. Having said that, by augmenting the suggestions with cryptography (and migrating to AML), one would have a very similar result to the one presented in this paper.

7.4 FMI as Open Document

The current FMI standard is very similar to the storage format described by the open document standard *IEC 29500* (2012). As the open documents standard has some security features embedded, this would allow to secure the FMUs directly. It might thus be considerable to make the FMI fully conforming with *IEC 29500* (2012), probably a rather small change. As this would require a change in the core FMI standard, it is neglected for this paper, though.

8 Prototype

In order to show feasibility of the proposed approach in Section 7.3 a prototype as a proof of concept has been implemented. This approach is a good tradeoff: it allows arbitrary extensions, is attached with products more and more, works offline, and is intentionally not backward compatible (which might cause security risks by false trust). The prototype does not carry out any real simulations but provides a way to execute and validate the complete process as described in this paper in a minimal environment.

The prototype has three major functions: first, in a bootstrapping process a self-signed PKI can be generated. This allows for local testing and understanding the concepts involved. Additionally, there is the option to sign an FMU and pack it into an AMLX file. This allows also to generate AMLX files with broken or spoofed FMUs to test the detection. Finally, one can extract the FMU again from the AMLX while checking the security measures.

The bootstrapping function allows to create a complete PKI from scratch with a self-signed root certificate as CA. Obviously, this should not be used for production but only serves as a demonstrator of the process. For more details, please have a look at the corresponding code and documentation (Christian Wolf 2023). This step will as well generate multiple AMLX files in accordance to the description in Section 7.3:

nominal A fully functional FMU, correctly hashed and signed

broken To simulate a defective file/transport, the FMU is modified but the other metadata are copied from the nominal case

tampered The FMU and its corresponding hash is modified in the AMLX by a targeted attack, all other metadata is copied over

The second step is to sign a custom FMU with the certificates as provided in the test instance. This process is rather straight forward, as no tests are carried out. Only the required files are read and interpreted and a valid signature is created. The process is depicted in Figure 2 as a flow diagram.

As final step, the prototype provides a way to check any AMLX file against a given root certificate. It allows to export the embedded FMU into a stand-alone file that can be run in legacy FMI-3-based simulation tools. Alternatively, it can serve as a boilerplate to implement an import feature for productive third party simulation tools.

DOI
10.3384/ecp204487

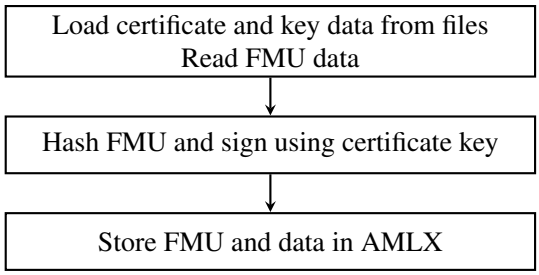

Figure 2. Flow diagram of the signing of FMUs.

The verification process is more complex than the signing as various checks need to be carried out. The ordering is in general of lower priority as all test must necessarily pass.

- (optional) The hash stored in the AMLX file must match the hash of the stored FMU.
- The certificate chain must be anchored on a user-provided trust anchor
- The complete certificate chain must be a chain, no branching is allowed and the chain must be in the correct ordering. Each certificate must be signed by the next certificate in the chain.
- Each certificate in the chain must not be expired.
- The signature must be valid

The overall process is sketched in Figure 3

Please note that the extraction process checks the certificates but after the splitting, the FMU is just a common FMU. There are no security features attached anymore. Changing the FMU after the export will not be detectable anymore in a secure manner. As a minimal measure to prevent accidential changes, a hash file is generated automatically during export.

By testing the various auto-generated examples in the first step through the checker, it is possible to see that the demonstrator can detect the changes in the FMU and metadata. Only the nominal version is accepted by the prototype.

9 Further Work

A survey with various industry partners is carried out at the time of writing. The goal is to identify detailed requirements and wishes from both FMU providers and consumers. Especially the position of the producers and their need to protect/encrypt the models is not yet analyzed strictly. Using this as a baseline, further improvements should be investigated as well as the other implementations in Section 7 addressed.

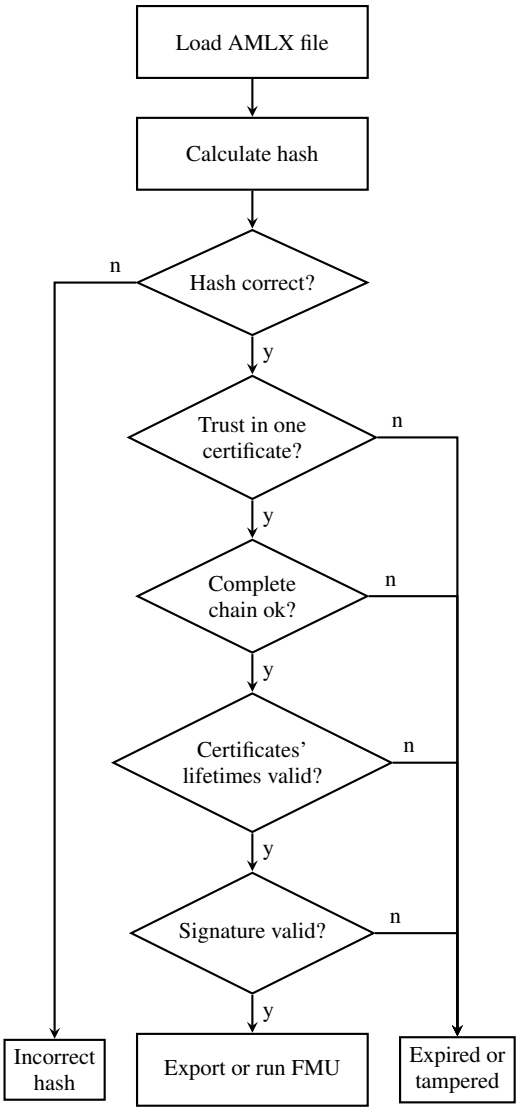

Figure 3. Flow diagram of verifying and exporting an FMU.

10 Summary

In this paper the general security problem especially for untested, unverified, or black-box simulation models is focused. There are different attack vectors presented that could be used to compromise the usage of shared models. Each attack vector has its individual attack surface and risk involved.

Due to the impossibility to prevent all attacks in general, an abstract analysis of the situation and possible general solutions are given. There are four possible implementations shown to realize the abstract considerations. For one of the four options a prototype has been implemented to show the effectiveness of the approach as a proof of concept.

The same approach as presented in this paper can be used to augment combinations of FMUs: one can use the System Structure and Parametrization standard (*SSP* 2022) instead of plain FMUs to be embedded.

Acknowledgements

The EKS contribution to this paper is part of the AIToC project. AIToC is a project of the European ITEA Initiative, funded by the German Federal Ministry of Education and Research under the funding code 01IS20073C.

References

Alex Biryukov, Dumitru-Daniel Dinu, and Dmitry Khovratovich (2015). *Argon and Argon2*. Password Hashing Competition (PHC).

Andre Bröring et al. (2022-11-03). "An Asset Administration Shell Version Control to Enforce Integrity Protection". In: *Kommunikation in Der Automation : Beiträge Des Jahreskolloquiums KommA 2022*. Vol. 13, pp. 192–203.

Avizienis, A. et al. (2004). "Basic Concepts and Taxonomy of Dependable and Secure Computing". In: *IEEE Transactions on Dependable and Secure Computing*. DOI: 10.1109/TDSC.2004.2.

Bart Preneel (1994). "Cryptographic Hash Functions". In: *European Transactions on Telecommunications* 5.4, pp. 431–448.

Blake C. Ramsdell (1999-06). *S/MIME Version 3 Message Specification*. DOI: 10.17487/RFC2633.

Christian Wolf (2023). *Christianlupus-Phd/Prototype-AMLX-checker*. URL: https://github.com/christianlupus-phd/Prototype-AMLX-checker.

Christian Wolf, Miriam Schleipen, and Georg Frey (2023-05-11). "Dynamische Systemmodelle austauschen mit FMI – aber sicher!" In: *atp magazin* 2023.5, p. 26.

Christoph Schlueter Langdon and Karsten Schweichhart (2022). "Data Spaces: First Applications in Mobility and Industry". In: *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*. Ed. by Boris Otto, Michael ten Hompel, and Stefan Wrobel, pp. 493–511. DOI: 10.1007/978-3-030-93975-5_30.

Daniel McCarney (2017-06). "A Tour of the Automatic Certificate Management Environment (ACME)". In: *The Internet Protocol Journal* 20.2, pp. 2–14.

Deutsches Institut für Normung, ed. (2019). *Asset Adminstration Shell Reading Guide*. 1st edition. ISBN: 978-3-8007-4990-4 978-3-410-28919-7.

IEC 63278-1 (2022-06-17). *DIN EN IEC 63278-1: Asset Administration Shell for Industrial Applications - Part 1: Administartion Shell Structure*.

Emin Topalovic et al. (2012). *Towards Short-Lived Certificates*.

FMI V3 (2023). *Functional Mock-up Interface Specification v3.0*. URL: https://fmi-standard.org/docs/3.0/ (visited on 2023-01-06).

Hedda Massoth (2022-11-21). *Catena-X Operating Model Whitepaper, Release V2*. URL: https://catena-x.net/fileadmin/user_upload/Publikationen_und_WhitePaper_des_Vereins/CX_Operating_Model_Whitepaper_02_12_22.pdf.

IEC 29500 (2012-09-01). *ISO/IEC 29500-2: Open Packaging Conventions*.

Jean-Philippe Aumasson et al. (2008). "Sha-3 Proposal Blake". In: *Submission to NIST* 92.

Johannes Mezger et al. (2011-07). "Protecting Know-How in Cross-Organisational Functional Mock-up by a Service-Oriented Approach with Trust Centres". In: *2011 9th IEEE International Conference on Industrial Informatics*, pp. 628–633. DOI: 10.1109/INDIN.2011.6034951.

Keith Eugene Maskus (2000). *Intellectual Property Rights in the Global Economy*. Peterson Institute.

Michael Klooß, Anja Lehmann, and Andy Rupp (2019). "(R)CCA Secure Updatable Encryption with Integrity Protection". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. DOI: 10.1007/978-3-030-17653-2_3.

Olaf Graeser et al. (2011). "AutomationML as a Basis for Offline and Realtime Simulation - Planning, Simulation and Diagnosis of Automation Systems:" in: *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics*, pp. 359–368. DOI: 10.5220/0003537403590368.

Rainer Drath (2021-07-19). *AutomationML: A Practical Guide*. Walter de Gruyter GmbH & Co KG. 290 pp. ISBN: 978-3-11-074623-5.

Rainer Drath, Markus Rentschler, and Michael Hoffmeister (2019-09). "The AutomationML Component Description in the Context of the Asset Administration Shell". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1278–1281. DOI: 10.1109/ETFA.2019.8869214.

Rajeev Sobti and G Geetha (2012). "Cryptographic Hash Functions: A Review". In: *International Journal of Computer Science Issues (IJCSI)* 9.2, p. 461.

Roberto Minerva, Gyu Myoung Lee, and Noel Crespi (2020-10). "Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models". In: *Proceedings of the IEEE* 108.10, pp. 1785–1824. DOI: 10.1109/JPROC.2020.2998530.

Ronald L. Rivest (1998). "Can We Eliminate Certificate Revocation Lists?" In: *Financial Cryptography*. Ed. by Rafael Hirchfeld. Red. by Gerhard Goos, Juris Hartmanis, and Jan Van Leeuwen. Vol. 1465, pp. 178–183. DOI: 10.1007/BFb0055482.

Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman (1978-02-01). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21.2, pp. 120–126. DOI: 10.1145/359340.359342.

Sharon Boeyen et al. (2008-05). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. DOI: 10.17487/RFC5280.

Simson Garfinkel (1995). *PGP: Pretty Good Privacy*. "O'Reilly Media, Inc." 442 pp. ISBN: 978-1-56592-098-9.

Stefan Tillich et al. (2009). "Compact Hardware Implementations of the SHA-3 Candidates ARIRANG, BLAKE, Grøstl, and Skein". In: *Cryptology ePrint Archive*.

SSP (2022-07-25). *System Structure and Parametrization V 1.0.1*.

Thomas Agrikola (2021). "On Foundations of Protecting Computations". PhD thesis. Karlsruher Institut für Technologie (KIT). 281 pp. DOI: 10.5445/IR/1000133798.