A Digital Twin Based Industrial Automation and Control System Security Architecture

Christian Gehrmann and Martin Gunnarsson

***Abstract*—The digital twin is a rather new industrial control and automation systems concept. While the approach so far has gained interest mainly due to capabilities to make advanced simulations and optimizations, recently the possibilities for enhanced security have got attention within the research community. In this paper, we discuss how a digital twin replication model and corresponding security architecture can be used to allow data sharing and control of security-critical processes. We identify design-driving security requirements for digital twin based data sharing and control. We show that the proposed state synchronization design meets the expected digital twin synchronization requirements and give a high level design and evaluation of other security components of the architecture. We also make performance evaluations of a proof of concept for protected software upgrade using the proposed digital twin design. Our new security framework provides a foundation for future research work in this promising new area.**

***Index Terms*—security, digital twin, state replication, security framework, security analysis**

I. Introduction

Industrial Automation and Control Systems (IACS) is a very broad term covering everything relating to control, monitoring and production in different industries and encompasses all parts of such systems.

While security for IACS in the past was neglected, in recent years security has obtained a lot of attention in the research community and indeed within the industry. Major security incidents such as the STUXNET worm in 2010 [1], the Shamoon Saudi Aramao spear-phishing attack in 2012 [2] and the German steel factory attack in 2014 [3] have highlighted the risk of attacks on IACS. Even if the attacks have been of many different types and origins, they have highlighted the need for enhanced security mechanisms and countermeasures.

Clear evidence that the industry nowadays takes security issues seriously is the development of best practice security guidelines [4] and the large number of security standards targeting the IACS domain, like ISO/IEC 27000 series[1], the ISA/IEC IEC 62443 series[2] and the NIST SP800 series. Among those, IEC 62443 is based on the very general ISO 27000 but specified for the IACS area and also the NIST SP 800-82 [5] in the SP800 series is an IACS standard. In addition, the industrial internet consortium has developed a new security framework [6].

New technology trends affect IACS as well as the entire society. Security solutions, security recommendations as well as standards, need to adapt to the new technologies. One clear current trend is the move from legacy ISA-95 to highly distributed and cloud based architectures according to the Industry 4.0 and RAMI 4.0 models [7]. This transition is demanding in many ways, one challenge is control and information sharing between production units and cloud based control functions. This constitutes a major security risk and requires careful system engineering not to jeopardize IACS reliability [8]. We tackle this general security issue in this paper by looking into the digital twin model as an *enabler* for enhanced security when opening up IACS low level control functions and data exchange according to the Industry 4.0 vision. Digital twins and state replication as security enablers were recently proposed by different researchers [9], [10], [11]. Previous works have not taken an IACS holistic view and in this paper we look into the problem from a system security point of view. The work is focused on identifying main design driving requirements for a digital twin based IACS security architecture and with special attention to a state synchronization model fulfilling the requirements. Detailed design of the different components and protocols in the architecture as well as formal security analysis of these are left for future work. The main contributions of the paper are the following:

- We introduce a digital twin IACS adversary model and identify security requirements for this model.
- We suggest a novel digital twin based security architecture including a new state replication model.
- We evaluate the security of the proposed state replication model as well as present a proof of concept implementation for a PLC software upgrade case including performance figures.

We proceed as follows: we discuss the digital twin model and make basic definitions which we use throughout the paper (§II), we introduce our adversary model and derive security requirements (§III), we suggest a new digital twin security architecture and a novel digital twin design, including a state replication model (§IV). We make a security analysis of the proposed model and architecture (§V) and present a proof of

Work supported by framework grant RIT17-0032 from the Swedish Foundation for Strategic Research as well as the EU H2020 project CloudiFacturing under grant 768892.

C, Gehrmann is with the Dept. of Electrical and Information Technology, Lund University, Box 118, 221 00 Lund, Sweden (email: christian.gehrmann@eit.lth.se). M. Gunnarsson is wih the Dept. of Electrical and Information Technology, Lund University, Box 118, 221 00 Lund, Sweden and also with RISE, Ole Römers väg 5A, 223 63 Lund, Sweden (email: martin.gunnarsson@ri.se)

[1]https://www.iso.org/isoiec-27001-information-security.html

[2]ISA, ISA99, Industrial Automation and Control Systems Security, https://www.isa.org/isa99/

concept implementation, including performance figures (§VI). Lastly we discuss related work (§II-B) and conclude (§VII).

II. Digital twin concept, related work and definitions

A. Digital twin model and scenario

The digital twin was according to Grives [12], a terminology invented around 15 years ago by John Vickers of NASA and the term was introduced publicly by NASA in 2010 [13]. Originally, the concept was used to refer to the digital representation of a product used in simulations software but has been expanded to a concept where not only a physical product is represented in virtual form (software) but each product is directly connected with a virtual counterpart, the digital twin. The general model is depicted in Fig. 1.

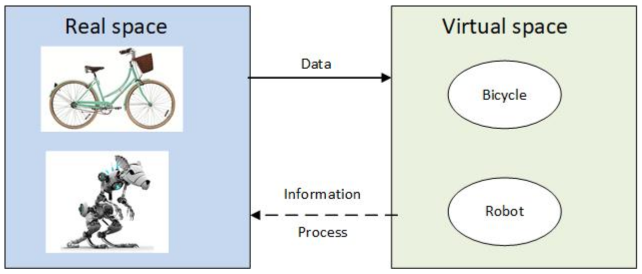

Fig. 1. The original digital twin model.

The overall goal with this concept is to be able to closely follow products during production (the physical twin) and simulate the process to adjust the production with results of these simulations. This can be done in real-time or close to real-time to optimize production flows etc. [14]. The concept has then been extended to include all units (robot loading stations, conveyor belts etc.) in a production system allowing advanced simulations of a complete manufacturing system and the units involved in an autonomous system [15]. Typically, then the digital twin part is represented and executed on cloud resources [16]. Fig. 2 illustrates the overall scenario and model.

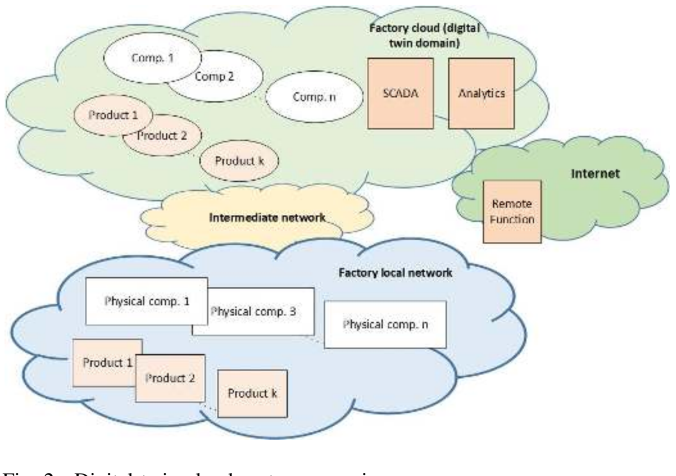

Fig. 2. Digital twin cloud system scenario.

As can be seen in Fig. 2, according to this model not only are the products themselves reflected as digital twins in the virtual (cloud) domain but also the manufacturing units or what we here refer to as ”components”. Typical components here are PLCs, historians, sensors, actuators data acquisition units, HMI units etc. Several different models and principles for reflecting such units are possible [11]. Here we focus on the network and logical state of a physical twin rather than the physical properties. The definitions and notations we use are introduced in Section II-C below.

B. Related work

Lots of work has been devoted to security in IACS. We will here briefly discuss literature surveys and how our architecture relates to the main security issues previously identified. Next, we will discuss some important previously introduced digital twin models and their relations to our approach. We mainly focus on prior work devoted to digital twin and state replication as enablers for enhanced security.

Security in IACS in general has been treated in several good surveys [17], [18]. The work by Krotifil and Gollmann [17] focusing on different types of attacks on existing systems but also concluding that most efforts so far have been devoted to IDS. Many existing IDS are compatible with our suggested architecture but it has the benefit that such systems can be deployed in the virtual domain. A very broad systematic overview of security in cyber-physical systems in general (including IACS) is given in by Humayed *et al.* in [19]. The authors identify that major security challenges in IACS are change management (including SW update) as well as the ability to handle legacy systems. Both these issues are tackles with the architecture we proposed in this paper. In addition, as we discussed in the introduction, several existing standards and new standard initiatives, are addressing IACS security in current and future systems. None of the main standardization bodies have so far been working with the digital twin concept as an enabler for enhanced security.

State machine replication has a very long history. Most of the work in this domain has been devoted to *fault tolerance* [20], [21]. Achieving state replication under the assumption of fault is much more demanding than the security oriented state replication we consider in this paper. We use a different, simpler model, allowing to choose the correct level of state reflection on the digital twin side depending on the security needs (see our state replication model in Section IV-B). This is justified by the fact that the design goal of a digital twin security system is disparate from a fault tolerance system, as the digital twin cannot replace the physical twin if it fails, but is there to reflect the physical twin and protect it from direct, potential hostile, external interactions.

The digital twin model was first introduced in [13]. Lots of work has then been devoted to the topic in resent years and good overview is given in [22]. The main focus has been on support of health analysis and improved maintenance as well as digitally mirroring the life of the physical entity. We are following the second approach but different from prior the majority of prior art, we are focusing on using the digital twin as an enabler for enhanced security.

The usage of digital twins for penetrations testing is discussed in a recent work by Bitton *et al.* [9]. The author investigate the relation behind a penetration test specification and system realization with focus on system cost optimization. A non-linear programming solution to find an optimal digital twin implementation level needed to perform certain security analysis tasks is presented. This is an approach that also is applicable to the sub-problem of digital twin realization in a system realizing the security architecture we present in this paper.

In [23] the idea of using state synchronization as an IoT security enabler was suggested. However, the model presented in [23] does not cover state changes on the IoT device side and no complete digital twin state synchronization model is given. Most recently, a digital twin security framework was presented in [11] and later extended in [10]. In [11], a digital twin specification principle using Automation ML (AML)[3] was described together with a proof of concept implementation detecting a man-in-the-middle PLC attack. In the follow up work, [10], also the state replication problem is considered. In this work, a passive state replication model is presented where state updates are purely done based on inputs in the physical domain. The strength with such a model is that it avoids the negative performance impacts of active state monitoring. Inspired by the work in [11] and [10], we have also looked into the problem area of state modeling as security enabler. However, different from the work in [10], we are looking into how digital twin can protect IACS from *external* attacks and not attacks on the factory domain. With this goal, we have proposed a different state propagation model and a security design allowing to identify attacks at the virtual domain and preventing them for even reaching the physical domain. Furthermore, we have analyzed a complete digital twin system scenario and proposed an overall security architecture for such scenario.

C. Digital twin definition and notations

For the purpose of the paper we denote by $u \in U$, a physical twin, where U denotes the set of physical twins in the system. Similarly, we denote by $u' \in U'$, a digital twin where U' is the set of digital twins in the system. Let then $S_u = \{s_{u0}, s_{u1}, ..., s_{um-1}\}$ and $S_{u'} = \{s_{u'0}, s_{u'1}, ..., s_{u'n-1}\}, m \geq n$, be the finite set of states of u and u', i.e., we assumes that the digital twin always only reflects a *subset* of the physical twin states and no states which are not represented in the physical twin. Furthermore, denote by $I_u = \{i_{u0}, i_{u1}, ..., i_{ur-1}\}$ the set of possible finite inputs to physical twin u and by $I_{u'} = \{i_{u'0}, i_{u'1}, ..., i_{u'd-1}\}$, the set of finite possible inputs to digital twin u'. We denote by $s_{u,t} \in S_u$, the state of physical twin u at clock cycle t and by $i_{u,t} \in I$ the input to u at clock cycle t. Similarly, denote by $s_{u',t} \in S_{u'}$, the state of digital twin u' at clock cycle t and the input to u' at clock cycle t by $i_{u',t} \in I'$. Hence, the initial state of the physical twin is $s_{u,0}$ and the initial state of the digital twin is $s_{u',0}$. Then we can define both the physical and digital twin as finite state machines. We then let $\delta_u : S_u \times I_u \to S_u$ and $\delta_{u'} : S_{u'} \times I_{u'} \to S_{u'}$ be the transition functions for the physical and digital twin respectively, i.e. $s_{u,t+1} = \delta_u(s_{u,t}, i_{u,t})$ and $s_{u',t+1} = \delta_{u'}(s_{u',t}, i_{u',t})$.

We assume a clock based digital twin state synchronization model where a each clock cycle, t, the state of the twins are synchronized with a message exchange starting with a first synchronization message from the u' to u and with a response synchronization message from u to u'. We denote these message as $m_{u' \to u}(t)$ and $m_{u \to u'}(t)$, respectively. These messages are typically not transferred in clear between the twin and intermediate nodes, but in protected/transformed form. We denote protected version of the synchronization messages by $e_{u' \to u}(t)$ and $e_{u \to u'}(t)$.

III. Adversary model and security requirements

Next, using the digital twin model and definition introduced in Section II, we describe a digital twin threat model. Using this threat model we identify security requirements for a digital twin based IACS architecture.

A. Adversary model

Adversary models for digital twin systems have not been extensively treated in the literature as the concept mostly so far has been used for production optimization and not security. Certain security aspects regarding using digital twin as security enablers in IACS are considered in [10] and [9]. The authors in [10] consider state replication for active monitoring and intrusion detection while [9] consider the problem of penentration testing of IACS with focus on cost optimization for specific security penetration tests (performed on simulated or emulated digital twin or on an acutal physical component in the IACS). However, since these works have very specific security functions goals, they lack adversary model definitions for the digital twin scenario we are considering. Hence, we have developed a new adversary model below. This is *not* a generic digital twin adversary model but a model that makes sense in systems with cloud based data sharing and control in IACS. We also give the main motivations for using this rather restrictive adversary model.

Traditionally, IACS has been separated with firewalls from other networks such as corporate network and the internet. Several good architectures and recommendations are available [6]. Here, we assume such principles are deployed and we have adopted an adversary model where we *do not* consider any attacks on the physical twin part or local factory network part of the system but assume these parts can be properly isolated from hostile external networks[4].

We assume that the digital twin can run in a separate process even on a third party cloud resource. Then the digital twin can be realized using virtualization techniques where the virtualization is offered on the most suitable level [25]. Providing strong isolation for virtualization and protection against hostile cloud providers is a very challenging topic which has been widely addressed with several different models and solutions the past ten years [26]–[28]. Recent attacks

[3]Actually, automation ML for digital twin modelling was already suggested by Grecyce *et al.* in 2016 [24] but not for any security applications.

[4]Internal factory network attacks are of course also possible, but we do not consider those in our adversary model.

Metldown [29] and Spectre [30] have shown that one cannot even trust the fundamental hardware functions needed for secure isolation currently in use. However, the security with respect to secure execution environment for virtualized systems is steadily improving and we will for simplicity in this paper disregard attacks on the isolation properties of the digital twin and assume that a secure execution environment and data storage is provided for the digital twin in the system.

We adopt the Dolev-Yao model [31] and assume that the attacker can influence the system in all other aspects including the following capabilities of the adversary:

- The attacker is able to intercept, modify and replay all communication from the physical domain to the digital domain and vice versa.
- The attacker is able to launch input attacks by sending arbitrary messages to a digital twin and input requests, i.e. he or she can choose to send arbitrary input from the set $I_{u'}$ to the digital twin u'.
- The attacker is able to launch intercept, modify and replay any information sent between digital twins or between digital twins and other units executing in the virtual domain.

B. Security definitions

Next, we give basic security definitions. The basis of the new security architecture is the introduction of state replication between the physical and digital twin. An expectation from such model from *robustness* perspective is that the synchronization is accurate over all system states and inputs. The synchronization consistent expectation is fundamental for deploying the architecture and very different from architectures introduced in the literature before. The main reason why consistency is important is that without it, one cannot rely on that all system changes in the digital twin part are correctly propagated to physical part of the system and vice versa, which will make it impossible to use the model in practise as the system behaviour would be unreliable. Hence, even if the synchronization consistency not is a pure security requirement, it is fundamental for the proposed architecture and we make a precise definition of synchronization consistent ency. It is also important to notice that one would expect from a specific design and implementation of our architecture to provide the synchronisation consistency property also under attack conditions. Hence, it is important to introduce a proper definition also in this regard.

Another fundamental, pure security expectation, with respect to the synchronization is the confidentiality and integrity of the synchronization process as such. Hence, we also provide precise definitions for these two aspects. Apart from these definitions, we adopt widely used computer and communication security definitions [32].

Definition III.1. A digital twin system is *consistent* if there exist functions $\forall u \in U, f_u : S_u \rightarrow S_{u'}$ such that the following is true:

$$\forall s \in S_u, f_u(\delta_u(s, \emptyset)) = \delta_{u'}(f_u(s), \emptyset), \quad (1)$$

$$s_{u',0} = f_u(s_{u,0}). \quad (2)$$

This definition reflects the requirement that when the digital twin starts in a state consistent with the staring state of the physical counterpart and whenever neither the physical twin nor the digital twin receive any input, they are both always transitioned to states that are consistent. i.e. the physical to digital twin state mapping agree with the state of the digital twin.

Definition III.2. A digital twin system synchronization protocol provides *confidentiality protection* if an adversary, who observes information, $e_{u' \rightarrow u}(t)$ and $e_{u \rightarrow u'}(t))$, sent from the digital twin and from the physical twin respectively at time t, cannot execute any attack, A, that in polynomial time will allow the attacker to distinguish the state of the physical twin from any randomly selected state, i.e., after execution of A, the following is true:

$$\forall s \in S_u, Pr(s_u = s | e_{u' \rightarrow u}(t), e_{u \rightarrow u'}(t)) = Pr(s_u = s) \quad (3)$$

Definition III.3. A digital twin system synchronization protocol provides *synchronization protection* if the adversary cannot execute any attack replacing message exchange $e_{u' \rightarrow u}(t)$ with $e'_{u' \rightarrow u'}(t)$ and/or replacing $e_{u \rightarrow u'}(t)$ with $e'_{u \rightarrow u'}(t)$ which will be accepted by u and u' and making the twins out of synchronization, i.e. $f_u(s_{u,t}) = s_{u',t}$ is always true after successful synchronization independent of adversary substitution choices[5].

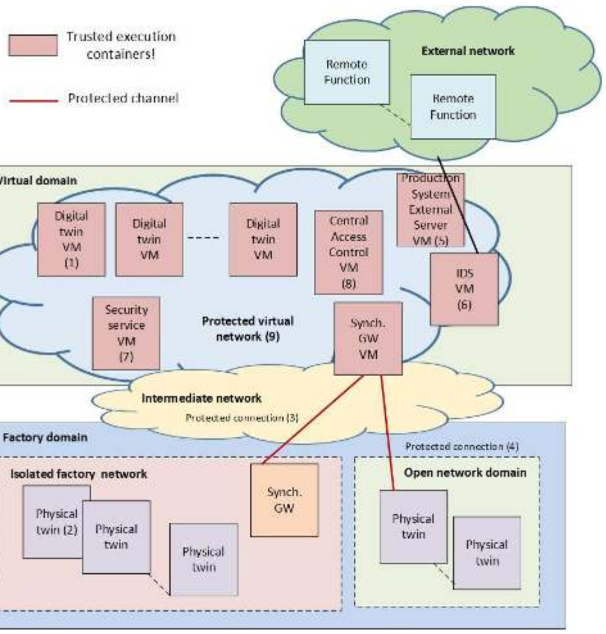

Fig. 3. Security architecture overview.

C. Requirements

We have used the previously presented adversary model and security definitions to identify a set of system security, performance and accuracy requirements. This is not an exhaustive list but the major identified system architecture requirements.

[5]This definition does *not* take a DoS attack into account and assumes that the synchronization messages arrives at each time slot.

R1. **Synchronization security:** We require the digital twin state replication model and protocol to be consistent (Definition III.1), provide confidentiality protection (Definition III.2) and synchronization protection (Definition III.3).
R2. **Synchronization latency:** The synchronization message exchange must not cause any delays which prevent time critical control functions to be propagated to from the physical to the digital twin. The precis requirements are application dependent.
R3. **Digital twin external connections protection:** All connections between the digital twin and the external entities must be authenticated. According to the adopted adversary model, we assume each digital twin to run in a protected execution environment but all request external to this environment must be properly authenticated and all information sent from the digital twin to external trusted parties must be confidentiality and integrity protected.
R4. **Access control:** The digital twin itself or a secure entity in direct connection with the digital twin needs to make sure access control is applied on on all incoming requests. This includes request and information exchange with external parties as well as information exchange with other digital twins.
R5. **Software security:** The physical twin software must always be in a trustworthy state. This implies that the physical twin must be protected from installation of harmful software. Mechanisms shall be in place to recover the system in case of zero-days attacks on the physical twins.
R6. **Local factory network isolation:** The local factory network shall not accept any connection requests except for protected synchronization requests with the digital twin (see R1 above). Physical twins should be protected from DoS attacks through boarder unit such as a gateway or firewall making sure that only protected synchronization requests reach a physical twin and no other outside traffic.
R7. **Digital twin Denial-of-Service (DoS) resilience:** The digital twin must be protected from DoS attacks such as network flooding or distributed DoS directly targeting a digital twin. Proper DoS filters and router configurations must be deployed in the factory cloud domain to prevent or limit the DoS possibilities of the attacker. At the same time, filters and router policies must not prevent synchronization exchanges to reach the digital twins in the system.

IV. A Digital Twin Based Security Architecture and State Replication Design

A. *Security architecture*

We now have the definitions and requirements in place to define a generic digital twin security architecture. Fig. 3 gives a high-level picture of the proposed architecture. We have here focused on the main security properties and entities in the system. This is *not* a complete design in all details but a high level design including main components and their roles in the architecture. We verify the key digital twin design of it in our proof of concept evaluation but leave detailed design and evaluation of other components for future work.

A basic security assumption in this architecture is the possibility to launch digital twins as well as security services in *trusted execution containers* as Virtual Machines (VMs) on suitable cloud resources. The architecture is completely agnostic on the virutalization technique used for this or on which actual level the vitalization is applied [25] [23]. However, the architecture requires the virtualization technology to provide trusted execution in the sense that different VMs are strongly isolated from each other and that they have access to protected volatile and non-volatile storage.

Using the numbering introduced in Fig. 3, we discuss the different properties of the components in the architecture below.

1) Digital twin component: The digital twin component is running as a VM in an isolated environment. An overview picture of the main logical functions of the twin is given in Fig. 4. The core functionality of the digital twin is the actual simulation of the physical counterpart. Only two direct external network interactions are allowed: the synchronization (which occurs over the synchronization GW) and the exchange with external requests and responses. This takes place either through the cloud server which takes *all* incoming requests and responses from external entities or directly to other digital twins or back-end components. The virtual domain external connections are protected through the cloud Virtual Private Network (VPN) (see Section IV-A9). The state of the digital twin is *exported* directly to a *common* (for several digital twins in a system) security analysis component (see also Subsection IV-A7). Also the intermediate state, $\hat{s}_{u'}$, is exposed to an analyzer in this way. This implies that an external analyzer can have access (if allowed by the access policy) to all digital twin states in the system. This in turn allows *abortion* of state propagation in case of detection of a fatal security issue by the external analyzer. The digital twin has access to a secure clock, t, for precise synchronization operations with the physical twin. The actual state propagation design we use is described in Section IV-B.

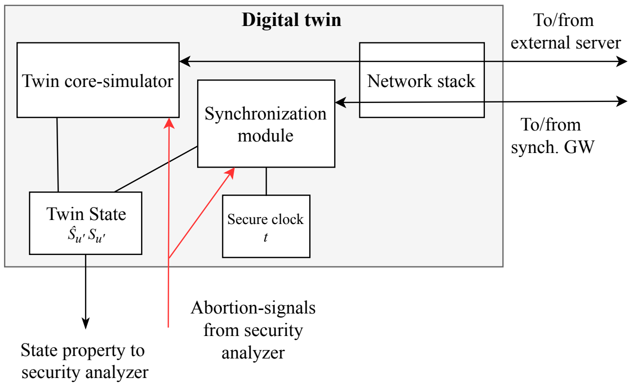

Fig. 4. Digital twin main functions

2) Physical twin component: An overview picture of the main logical functions of the physical twin is given in Fig. 5. Similar to the digital counterpart, the physical twin executes the defined synchronization protocol. Depending on if the physical twin actually has network connectivity or not, it might run the synchronization itself or it is done through a ”measurement unit”[6]. A physical twin deployed in an isolated factory network will only exchange synchronization information with a dedicated synchronization GW on the same network. On the other hand, a single deployed physical twin outside such a network will need to directly exchange synchronization information with the synchronization GW in the virtual domain and needs access to the key material needed for such secure interactions. The physical twin will apart from this, not need any specific security adaptations at all. The state propagation design applicable to the physical twin is described in Section IV-B.

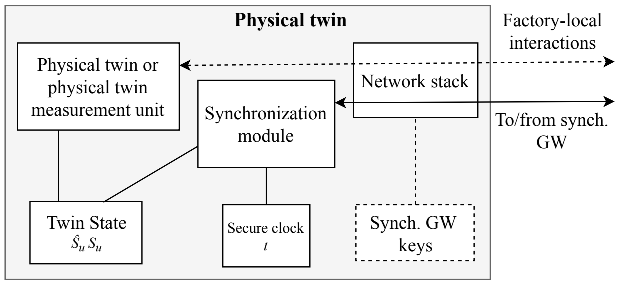
Physical twin
Factory-local interactions
Physical twin or physical twin measurement unit
Network stack
Synchronization module
To/from synch. GW
Twin State $\hat{S}_u\, S_u$
Secure clock t
Synch. GW keys

Fig. 5. Physical twin main functions.

3) Protected connection between synchronization gateways: The connection between the synchronization GW on the local factory and the virtual domain is protected through a secure channel. We have chosen this principle instead of end-to end synchronization protection as we assume it will be possible to deploy synchronization GWs in trusted containers in the virtual domain. Standard IPsec [33] VPN or a TLS/DTL channels [34] [35] are assumed. A major advantage with such solution from security management point of view is that this allows a *single* security relation between the physical and digital domain. Such single relation is very easy to maintain from security perspective. For instance, can a pres-shared key TLS or DTLS relation for instance be used. This can be compared to a situation where external entities are allowed to directly connect to the physical domain. In such situation, each external connection would need a separate security relation with the physical domain. Now, such relations are instead moved to the digital domain, where the security risk is much lower and where it is much less complex to handle such relations from a security configuration management point of view.

4) Protected connection from isolated physical twin to synchronization gateway: A physical twin not deployed in a protected local factory network, needs to directly connect to the synchronization GW in the virtual domain. This connection then obviously needs to be confidentiality and integrity protected using a suitable secure channel (see Section IV-A3).

5) Production system external server: The architecture assumes all external requests arrives in the virtual domain, i.e. external input to digital twin u' from the set $I_{u'}$ arrives to the production system external server prior to (potential) being forwarded to the digital twin u'. Similar responses from a digital twin are routed through this sever as well. This allows advanced network filtering at a single point and avoids having such functionality duplicated at each digital twin virtual instance[7].

6) Intrusion Detection System (IDS): State-of-the art IDS are best deployed at the boarder to the internet [36]. We adopt this principle and assume the actual intrusion analysis to be done by a VM with direct access to the external network interface traffic.

7) Security analysis service: The core benefit from a security perspective with a digital twin model like the one we have defined, is the possibility to do security analysis directly on the digital twin state and even on the states of a whole family of digital twins. By letting the analyzing engine having access not only to the final states, but also intermediate states, i.e. the $\hat{s}_{u'}$ states in the system, it is possible for a security analysis function to detect harmful state transitions (prior to the state propagating to the physical twin) and take direct action in the digital domain (see also Fig. 4).

8) Central access control: By letting all external digital twin access be subject to a single point access control, system wide policies can easily be deployed in the system. Advanced security policies can be defined through standard access control frameworks such as Extensible Access Control Markup Language (XACML) [37]. In order to allow direct interaction between digital twins, this is preferably combined with component local policy enforcement through tokens issued at the central access control entity using standard tokens such as SAML [38] or OAuth [39].

9) Protected virtual network: Most cloud providers offer network isolation between VMs launched on cloud resources[8][9]. Even if we have not assumed all trusted execution services to be deployed as complete, ”traditional” VMs in the virtual domain, higher layer VMs can be launched on such VMs allowing re-use of standard principles for network isolation. There are also other, non-provider dependent solutions to achieve this [40].

B. State replication model and design

Several different state replication principles for digital twins are possible. Recently, a specification-based state replication model for digital twins was proposed [10]. We have adopted a similar physical and digital twin state transition model. However, the state replication design in [10] is built upon measurement of input values and that the physical and digital twin runs *functional identical programs* or what the authors refers to as ”passive state replication”. This is an approach that is efficient if the main purpose of the design is to evaluate security breaches stemming from the physical domain. Instead,

[6]For a physical twin that is in production, it could be that it has no program execution capabilities, but its state is only measured through external sensors for instance.

[7]Recall that in our adversary model we assume all inputs to a *physical* twin to be trustworthy and not subject to direct security analysis

[8]https://docs.aws.amazon.com/vpc/index.html#lang/en_us

[9]https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

we in our security architecture use the digital twin as a ”guard” against all, potential hostile, *external* stimuli on the physical domain. Hence, even if demanding from real-time perspective, we instead have adopted a direct state replication or what the authors in [10] refers to as ”active monitoring”. This different security goal and approach also allow us to abandon the functional identical program requirements. We assume a model, where the physical and digital twin are synchronized on regular basis. Without loss of generality, we assume that a synchronization is done at each clock cycle. Let $z_u : S_u \times S_{u'} \to S_u$ be a synchronization function and $h_u : S_u \to S_{u'}$ a physical to digital state mapping function for twin u. The complete synchronization (including the twins state updates) then consists of the following operations:

$$\begin{aligned} \hat{s}_{u,t+1} &= \delta_u(s_{u,t}, i_{u,t}), && (4)\\ \hat{s}_{u',t+1} &= \delta_{u'}(s_{u',t}, i_{u',t}), && (5)\\ s_{u,t+1} &= z_u(\hat{s}_{u,t+1}, \hat{s}_{u',t+1}), && (6)\\ s_{u',t+1} &= h_u(s_{u,t+1}). && (7) \end{aligned}$$

This synchronization model works such that the physical and digital twin treat their respective inputs independently. We assume that the input will change the state of the (respective) twins independently, and then at the next time slot, they will synchronize their states to make them consistent considering the inputs received before last synchronization.

The choice of the functions z_u and h_u will depend on the digital twin model and the exact relation between the physical and digital twin. Many different models are possible. For the purpose of this paper, we choose a simple twin model but still a model allow to cover several important security cases as we show in Section VI. Denote by $S_u = S1_u \bigcup S2_u \bigcup S3_u$, we then make the following assumption: $S1_u \bigcap S2_u = S1_u \bigcap S3_u = S2_u \bigcap S3_u = \emptyset$. Let $S_{u'} = S1_{u'} \bigcup S2_{u'}$ and we assume that $S1_{u'} \bigcap S2_{u'} = \emptyset$. Then we can write the state of the physical twin as $s_u = (s1_u, s2_u, s3_u)$ and the state of the digital twin as $s_{u'} = (s1_{u'}, s2_{u'})$. We then apply the following restrictions:

$$\begin{aligned} S2_u &= S1_{u'}, && (8)\\ S3_u &= S2_{u'}, && (9) \end{aligned}$$

$$\begin{aligned} &\forall s_u \in S_u, \forall i_u \in I_u, \delta_u(s_u, i_u) = \\ &= (\delta 1_u(s_u, i_u), \delta 2_u((s2_u, s3_u), i_u), s3_u), && (10)\\ &\forall s_{u'} \in S_{u'}, \forall i_{u'} \in I_{u'}, \delta_{u'}(s_{u'}, i_{u'}) = \\ &= (s1_{u'}, \delta 2_{u'}(s_{u'}, i_{u'})) && (11) \end{aligned}$$

In addition, we let

$$s_{u',0} = (s1_{u',0}, s2_{u',0}) = (s2_{u,0}, s2_{u',0}), \quad (12)$$

$$s_{u,0} = (s1_{u,0}, s2_{u,0}, s3_{u,0}) = (s1_{u,0}, s2_{u,0}, s2_{u',0}) \quad (13)$$

With these restrictions, we then let $z_u(\hat{s}_{u,t}, \hat{s}_{u',t}) = (\hat{s}1_{u,t}, \hat{s}2_{u,t}, \hat{s}2_{u',t})$ and

$$h_u(s_{u,t+1}) = \begin{cases} (s2_{u,0}, s3_{u,0}) & \text{if } t < 0 \\ (\hat{s}2_{u,t+1}, \hat{s}2_{u',t+1}) & \text{otherwise} \end{cases} \quad (14)$$

To send the *complete* state at each synchronization occasion is very inefficient. Instead, the state changes (deltas) are calculated:

$$\begin{aligned} m_{u'\to u}(t) &= \Delta_{\hat{s}_{u'}} = \text{Diff}(\hat{s}_{u',t+1}, s_{u',t}), && (15)\\ m_{u\to u'}(t) &= \Delta_{s_{u'}} = \text{Diff}(\hat{s}2_{u,t+1}, s2_{u,t}), && (16) \end{aligned}$$

This implies that the digital twin calculates a first delta, $\Delta_{\hat{s}_{u'}}$, and sends it to the physical twin. This delta is then used by the physical twin to reconstruct $\hat{s}_{u',t+1}$, which is the input to the z function, i.e. equation (6). Next, the physical twin calculates the ”return delta”, $\Delta_{s_{u'}}$, that is sent back to the digital twin. The principle is illustrated in Fig. 6 below. Observe, that we here only illustrate the synchronization information exchange and not the protection of the synchronization messages as such. The protection principles we apply was described in Section IV-A.

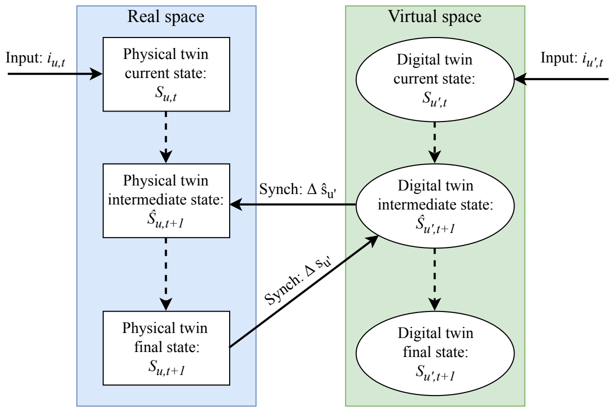
Real space
Virtual space
Input: $i_{u,t}$
Physical twin current state: $S_{u,t}$
Physical twin intermediate state: $\hat{S}_{u,t+1}$
Physical twin final state: $S_{u,t+1}$
Synch: $\Delta\ \hat{s}_{u'}$
Synch: $\Delta\ s_{u'}$
Digital twin current state: $S_{u',t}$
Digital twin intermediate state: $\hat{S}_{u',t+1}$
Digital twin final state: $S_{u',t+1}$
Input: $i_{u',t}$

Fig. 6. Synchronization principle.

It is important to notice from real-time and communication overhead perspectives that when no input is received neither on the physical or digital side, there is no need for the twins to exchange any deltas. This is true given a consistent digital twin system synchronized with accurate clocks.

V. Security analysis

Next, we analyze the proposed framework from security and performance perspectives. We here mainly focus on the synchronization security characteristics. We also give arguments regarding how the proposed architecture meets the other security requirements listed in Section III-C. As the architecture in many aspects only include a high level design, we here postpone detailed security evaluation of these aspects to future work and for specific implementation designs.

1) Synchronization security:

Proposition 1. The digital twin synchronization model and protocol is consistent.

Proof. Let:

$$f_u(s) = f_u((s1, s2, s3)) = (s2, s3). \quad (17)$$

From (13) we have that $s_{u,0} = (s1_{u,0}, s2_{u,0}, s2_{u',0})$ and from (12) and (14), it then follows that $f_u(s_{u,0}) = h_u(s_{u,0}) =$

$(s2_{u,0}, s3_{u,0}) = (s1_{u',0}, s2_{u',0}) = s_{u',0}$, which fulfils condition (2).

Now, using the assumptions (8), (9), (10) and (11), let: $\hat{\delta}_u(s_u, i_u) = (\delta 1_u(s_u, i_u), \delta 2_u((s2_u, s3_u), i_u), \delta 2_{u'}((s2_u, s3_u), \emptyset)$. Then, it follows from (17), $f_u(\hat{\delta}_u(s_u, \emptyset)) = (\delta 2_u((s2_u, s3_u), \emptyset), \delta 2_{u'}((s2_u, s3_u, \emptyset))$. Similar, let: $\hat{\delta}_{u'}(s_{u'}, i_{u'}) = (\delta 2_u((s1_{u'}, s2_{u'}), \emptyset), \delta 2_{u'}(s_{u'}, i_{u'}))$. Then by direct calculation: $\hat{\delta}_{u'}(f_u(s_u), \emptyset) = \hat{\delta}_{u'}((s2_u, s3_u), \emptyset) = (\delta 2_u((s2_u, s3_u), \emptyset), \delta 2_{u'}((s2_u, s3_u), \emptyset)) = f_u(\hat{\delta}(s_u, \emptyset))$. By then letting the state s_u taking any value in S_u, it follows that also condition (1) is fulfilled. □

Proposition 2. If the secure channel used for communication towards and between synchronization GW in the architecture provides confidentiality, the digital twin synchronization design also provides confidentiality.

Proof. According to our attacker model, an adversary can intercept any message sent from the digital twin to the synchronization GW in the virtual domain or any messages sent between synchronization GWs. He or she might also intercept message sent from physical twins towards the GW deployed in the virtual domain. The attacker has no other option to intercept any synchronization information. According to (15) and (16), at each clock cycle, one delta message is sent from the digital twin towards the physical twin and a replay delta message is sent in return. An adversary has two options to intercept the first message, $e_{u' \to u}(t)$; Either he or she intercept it when it is sent from the digital twin the synchronization GW in the virtual domain *or* when it is forwarded from the synchronization to the GW in the factory domain (or physical twin in the second option). As long as both these channels provide confidentiality the attacker will not get any information on s_u. As the return message follows the very same path, the also the return message, $e_{u \to u'}(t)$, will have the very same protection and equation (3) is fulfilled. □

Proposition 3. If the secure channel used for communication towards and between synchronization GW in the architecture provides integrity and replay protection, the digital twin synchronization design also provides synchronization protection.

Proof. According to Proposition 1 the proposed synchronization model is consistent and consequently if no input is received on neither the digital nor physical twin, $h_u(s_{u,t+1}) = s_{u',t+1}$. Furthermore, if the synchronization messages also arrives unmodified equation (7) guarantees that $h_u(s_{u,t+1}) = s_{u',t+1}$ holds also in this case. Hence, the only option for an attacker would be to modify any messages $e_{u' \to u}(t)$ or $e_{u \to u'}(t))$. In analogue with the proof of Proposition 2, if the used secure channels provides integrity and replay protection, such modification will be detected and a modified or replayed message will be rejected. □

2) Latency: The architecture as such does not make any direct assumption regarding the synchronization real-time behaviour. Depending on the specific IACS application, the networks must be chosen and configured accordingly. Similarly, the synchronization GW must be implemented on platforms powerful enough to fulfill real-time requirements. For some applications, deploying the virtual domain on an edge cloud [8] can be used to meet R2.

3) External connections: The architecture assumes all external connection to be intermediates by the external server entity at the boarder of the external network. The external server will only accept authenticated requests. Furthermore, the final hop for the external server to the digital twin runs through the virtual domain VPN. This, if properly implemented, implies that the system fulfills the requirement R3.

4) Access control: According to the proposed security architecture, the centralized access control VM deployed in the virtual domain makes sure all access requests towards the digital twin are properly authorized. Access control enforcement then takes place at the digital twin VM. This means that the main building blocks are included to fulfil R4. However, the actually authorization and access control mechanisms which are supported are subject to detailed design, which have been left for future work.

5) Software security: The software state of the physical twin can be replicated to the digital counterpart. A security service with direct access to the twin state can be launched. This service then controls the physical twin software state and upgrade. This is a very efficient way to both monitor the SW status and control upgrades as we show with the experimental evaluation in Section Section VI. Even if this is an important step to meet R5, further SW monitoring tools needs to be deployed in the system to give the wanted software security level.

6) Network isolation and DoS resilience: The architecture adopts best practise for factory network isolation [5] to meet R6. In addition, external interaction with the factory domain is only possible indirectly through the protected synchronization. All direct requests towards digital twin are subject to IDS and filtering and additional security protection mechanism can be launched as security service VMs in the virtual domain. With proper design and implementation, such measures will provide network isolation and DoS resilience as required by R7.

VI. Proof of concept and performance evaluation

In order to test the feasibility of the proposed architecture and approach, we have implemented a low complexity system with digital twins using our proposed state synchronization protocol. Our main goal here is to get an impression of how the proposed synchronization framework, which is the fundamental basis of the proposed architecture, affects the production units in the system as well as the bandwidth consumption[10]. It was argued in [10] that direct state synchronization or what the authors refer to as active monitoring is not feasible in real-time critical systems due to large bandwidth overhead. While we argue that this is not the case for low complexity digital twin state models and for moderate synchronization frequencies, we are interested to measure the production unit actual computation and bandwidth overhead in a real system.

[10]We recall that the synchronization including the protection of the synchronization is the only parts of the architecture that directly affects the production domain.

To make the evaluation feasible, we here focus on the *first three* components in the architecture in Fig. 3. We have implemented a simple manufacturing scenario, as seen in Fig. 7 consisting of a PLC unit, u_1, controlling an industrial process. In addition, we have a software upgrade server, u_2, holding software upgrade information, that is deployed in the factory local network. The PLC and the upgrade server are reflected as digital twins: u'_1, u'_2. The goal with introducing the virtual domain is to allow secure software control and upgrade of the production system units. To facilitate this, the software state and software control state are replicated to the digital twin domain.

It should be noted, that additional components and more complex production scenarios, will give a more detailed picture of how the proposed synchronization model effects the system performance. However, as the proposed synchronization protocol scales linear with the number of units with respect to bandwidth consumption, we argue that measurements in a small systems will give a good view of the overall system impact. Furthermore, the actual effect in terms of computational overhead on a particular production unit, will obvious depend on the computational power of the unit. Here, we use a fairly constrained platform, a RaspberryPI, for the evaluation. Other platforms and systems will be affected in similar ways but obviously platforms with less resources will be affected more. How, different platforms with different resources are affected, is left for future work as our main goal here is to verify the general feasibility of the approach.

Our proof-of concept implementation shows that as long as we have moderate state changes and the synchronization happens less than 100 times a second, clock synchronization is not an issue. The platform we have worked with can timely process a request and send a response without major delays. Hence there is no need to have a more precise clock synchronization. Here we let the digital twin act as a "master" and the physical twin as a "slave" unit at each synchronization occasion.

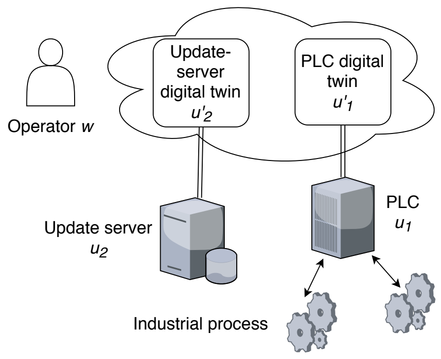

Fig. 7. Setup of out digital twin and software update scenario.

The state information for the supported twins are selected to be: $s_{u'_1}$ = [ctrl_flag, ctrl_url, sw_state] and $s_{u'_2}$ = [ctrl_url, sw_package][11]. ctrl_flag is a value holding software upgrade request control and error information and the ctrl_url is a URL of a new software package to be installed. sw_state is a list of all current software packages and versions installed on a unit and sw_package is a new software package. We also assumes a remote operator, w, to be present in the system controlling software upgrades through a remote user device over standard internet.

[11]Here is actually no state information with origin from the physical twin, u_2, but just digital twin state information which is propagated to the physical twin.

A. PLC software update process

w identifies a new software package, q, and connects to the external server u'_2. w then downloads q to u'_2 and w receives a ctrl_url value for the package in return. u'_2 then updates the state $\hat{s}_{u'_2,0}$ to reflect the storage of the new software package. Then a synchronization takes place between u'_2 and u_2. The synchronization is done by sending $\Delta_{\hat{s}_{u'_2}} =$ ctrl_url$+q$ from u'_2 to u_2. This in turn, triggers u_2 through the functions h_{u_2} and z_{u_2}, to update its internal state, resulting in the storage of q which can be downloaded from ctrl_url to other units within the local factory network.

w makes a second request using the newly received ctrl_url and with information regarding the new software packages towards u'_1. The request trigger u'_1 to update states $s_{u'_1,1}$: ctrl_flag, ctrl_url, sw_state, where ctrl_flag contains "available software update indicator", ctrl_url contains the URL to the new software package on u_2 and sw_state contains version information for the pending new software. In the clock cycle 2, this information is propagated to u_1 through $\Delta_{\hat{s}_{u'_1}}$. This values in combination with the functions h_{u_1} and z_{u_1} give an updated state $s_{u_1,2}$. The SW update flag in state $s_{u_1,2}$ triggers u_1 to set the state to update pending allowing to u_2 using ctrl_url to download and install the new SW package, q. Once, the update is finalized, the update status information as well as the new SW state information is propagated back to u'_1 through updates of the ctrl_flag and sw_state.

B. Performance evaluation

We have implemented the scenario, described above, with a SW update process using digital twins. As the PLC u_1 we have used OpenPLC [41], a free, open source PLC implementation, running on a RaspberryPI[12]. The Raspberry Pi we have used is a model 2 v1.1 with an ARM Cortex-A7 quad-core processor, clocked at 900MHz.

The digital twins u'_1, u'_2 are running as separate processes in a Ubuntu 18.04 desktop host. The same host also functions as the update server u_2. Since the physical entities synchronize with digital-twins outside the protected factory network the synchronization protocol is secured by DLTS.

1) Update time depending on synchronization frequency: In order to evaluate the state synchronization protocol we have looked at the SW update scenario. We want to examine how the state synchronization process affects other processes running on the system.

First we ran tests without state synchronization to establish a base line for how long time the update process takes. Then

[12]https://www.raspberrypi.org

we ran the SW update process with state synchronization at different frequencies. We evaluated performance at 1, 10 and 100 state synchronizations per second. The result can be seen in Fig. 8[13].

As can be seen from the figure the performance impact of the state synchronization is very small. Only at a large number of synchronizations per second is the performance noticeable.

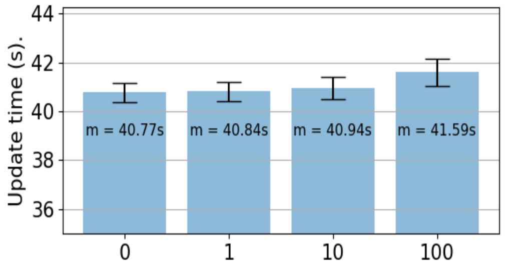

Fig. 8. Update times when using state synchronization at different frequencies.

2) *Compassion of DTLS Cipher Suites:* We have compared different DLTS cipher suites to evaluate if this impacts performance. The default strong suite AES-256-GCM with SHA384 was compared to the weaker AES-128-GCM with SHA256. The results can be seen in Figure 9. It can be noted that the choice of ciphers has only a very small impact on the performance of the update process.

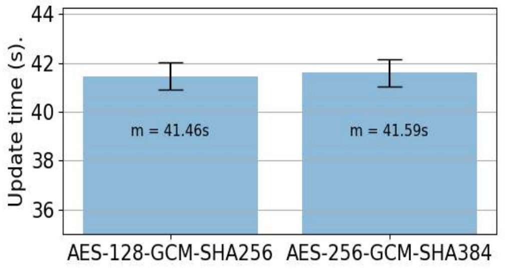

Fig. 9. Comparison of update times with different DTLS cipher-suites.

3) *Computation cost:* A PLC is not a constrained device in a traditional sense, however, since it controls a time-critical process CPU-time is limited. Any added features must consider this so time-critical deadlines are kept.

We have measured the CPU-time needed by the PLC to implement our state synchronization protocol. By running the protocol over an extended time we have come to the following numbers as seen in Table I.

As shown in the table the CPU-time needed by the PLC to implement the state synchronization protocol is very small. An even slower CPU will still be able to run the state synchronization without overloading the processor.

4) *Network performance:* Evaluating network performance for the state synchronization process is difficult to do without real ICS network traffic to base an evaluation scenario on. Hence, instead we evaluated the performance in an isolated system. We measured the bandwidth consumption for the PLC during the update process. We then measured the bandwidth for the update process while synchronizing with the PLC's digital-twin. The synchronization messages were of size 22 bytes in each direction. The bandwidth consumption can be seen in Table II. As can be seen from the Table the bandwidth consumption is reasonable for small synchronization frequencies.

[13]In the simple system we are using, actually, the state exchange can be omitted in most cases as we very seldom have state changes, but in our evaluation, we anyway forced a state exchange to take place in order to test the synchronization frequency performance impact.

CPU-time (ms) per synchronization	CPU-load 10 synchronizations/s	CPU-load 100 synchronizations/s
0.3772 ($s = 0.0602$)	0.0038%	0.0377%

TABLE I
MEASUREMENTS OF CPU-TIME PER STATE SYNCHRONIZATION MESSAGE AND CPU-LOAD.

	Bandwidth to PLC	Bandwidth from PLC
No synch	0.97 KB/s	2.06 KB/s
1 synch/s	1.20 KB/s	2.38 KB/s
10 synch/s	2.16 KB/s	3.35 KB/s
100 synch/s	10.88 KB/s	12.06 KB/s

TABLE II
BANDWIDTH TO AND FROM THE PLC WHEN UPDATING.

VII. CONCLUSION AND FUTURE WORK

Motivated by the need for new security models and principles in IACS to open up the systems for cloud based processing and data sharing, we investigated how digital twins can work as a security enablers in IACS. We introduced a new adversary model, made basic security definitions, identified security requirements, made a novel security architecture and in particular state replication design for a digital twin based IACS. The new state replication design as well as the architecture were then security evaluated against the identified requirements. We showed that the proposed synchronization design meets the introduced digital twin synchronization requirements. Furthermore, we made a high-level design of the other security components in the architecture and argue about how the suggested functions will help in meeting the identified security requirements. Through our proof of concept implementation and performance evaluation, we also showed that the new digital twin synchronization model works well in practice for a small but real production case with reasonable performance impact. Especially, we show that as long as we have not too high update frequency, the performance impact on a platform like RaspberryPI is negligible. As expected, the bandwidth increases linear with the synchronization frequency. In our evaluation, we only reflected a few PLC states, and obviously, the more fine grain states that are reflected, the more impact it will have on the system performance and bandwidth consumption.

The results shows that a digital twin based security architecture can be a promising way to protect IACS while open them up for external data sharing and access. We have here worked with defining a suitable overall architecture and synchronization model. In order to develop a fully working system based on our architecture and approach, more work is needed. Below, we discuss the most important future work:

- **Performance:** We have here made first proof of concept of the architecture. In order to see the effect of the architecture on different platform and production scenarios, more performance evaluations on different platform, with more complex digital twin state models and with larger amount of production nodes are needed.

- **Intrusion detection:** In our security architecture, we have only show how on principle level how to integrate intrusion detection at the boarder to the virtual domain. It is left for future research to design and integrate intrusion detection in a fully working system.
- **Access control:** The architecture allows for advanced access control in the virtual domain. The main advantage with this approach is that this can be supported without affecting the production domain at all. It remains to design and evaluate this approach in a full system implementation of the architecture.
- **Formal security analysis:** We have proven the consistency of the proposed synchronization protocol and showed that the security of the protocol depends on the security of the underlying used secure channel. Formal analysis of the security of the complete system design and all protocols are left for future work.
- **Security analysis services:** Apart from IDS and access control enforcement in the virtual domain, additional security analysis services may be supported as virtual components as we showed in our architecture design. This include services such as virus scan, DoS prevention etc. The design and evaluation of such services is left to future research as well.

Acknowledgement

We would like to thank the SSF SEC4FACTORY project team for valuable discussions regarding the research direction and results presented in the paper. In particular we would like to thank the TetraPak project members for their valuable feedback and suggestions.

References

[1] N. Falliere, Murchu, and E. Chien, "W32.Stuxnet Dossier," Symantec Security Response online report, Feb. 2011. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

[2] A. Leyden, "Hack on saudi aramco hit 30,000 workstations, oil firm admits," 2012. [Online]. Available: http://www.theregister.co.uk/2012/08/29/saudi_aramco_malware_attack_analysis/

[3] P. F. Roberts, "Cyberattack inflicts massive damage on german steel factory," The security ledger, 2014. [Online]. Available: https://securityledger.com/2014/12/cyberattack-inflicts-massive-damage-on-german-steel-factory/

[4] F. M. P. Didier P, J. Harstad *et al.*, "Converged plantwide ethernet solution - converged plantwide ethernet (cpwe) design implementation guide," Cisco Systems and Rockwell Automation, 2011. [Online]. Available: https://literature.rockwellautomation.com/idc/groups/literature/documents /td/enet-td001_-en-p.pdf

[5] K. Stouffer, V. Pillitteri, S. L. Marshall, and A. A. Hahn, "Guide to industrial control systems (ics) security," NIST Special Publication 800-82, 2, Version 2, 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-82r2.pdf

[6] S. Schrecker *et al.*, "The industrial internet of things - volume g4: Security framework," Industrial Internet Consortium, 2016. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB-3.pdf

[7] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *SME Manufacturing Letters*, vol. 3, 12 2014.

[8] J. Delsing, "Local cloud internet of things automation: Technology and business model features of distributed internet of things automation solutions," *IEEE Industrial Electronics Magazine*, vol. 11, no. 4, pp. 8–21, Dec 2017.

[9] R. Bitton, T. Gluck, O. Stan, M. Inokuchi, Y. Ohta, Y. Yamada, T. Yagyu, Y. Elovici, and A. Shabtai, "Deriving a cost-effective digital twin of an ics to facilitate security evaluation," in *Computer Security*, J. Lopez, J. Zhou, and M. Soriano, Eds. Cham: Springer International Publishing, 2018, pp. 533–554.

[10] M. Eckhart and A. Ekelhart, "A specification-based state replication approach for digital twins," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. ACM, 2018, pp. 36–47. [Online]. Available: http://doi.acm.org/10.1145/3264888.3264892

[11] ——, "Towards security-aware virtual environments for digital twins," in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, ser. CPSS '18. New York, NY, USA: ACM, 2018, pp. 61–72. [Online]. Available: http://doi.acm.org/10.1145/3198458.3198464

[12] M. Grieves, "Digital twin manufacturing excellence through virtual factory replication," Dassault Syst'emes, Paris, France, 2014. [Online]. Available: http://innovate.fit.edu/plm/documents/doc_mgr/912/1411.0_Digital_Twin_White_Paper_Dr_Grieves.pdf

[13] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," National Aeronautics and Space Administration (NASA), 2010. [Online]. Available: https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf

[14] T. H.-J. Uhlemann, C. Schock, C. Lehmann, S. Freiberger, and R. Steinhilper, "The digital twin: Demonstrating the potential of real time data acquisition in production systems," *Procedia Manufacturing*, vol. 9, pp. 113 – 120, 2017, 7th Conference on Learning Factories, CLF 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2351978917301610

[15] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567 – 572, 2015, 15th IFAC Symposium onInformation Control Problems inManufacturing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896315003808

[16] M. R. Shahriar, S. M. N. A. Sunny, X. Liu, M. C. Leu, L. Hu, and N. Nguyen, "Mtcomm based virtualization and integration of physical machine operations with digital-twins in cyber-physical manufacturing cloud," in *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, June 2018, pp. 46–51.

[17] M. Krotofil and D. Gollmann, "Industrial control systems security: What is happening?" in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, July 2013, pp. 670–675.

[18] P. Uchenna, D. Ani, H. M. He, and A. Tiwar, "Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective," *Journal of Cyber Security Technology*, vol. 1, no. 1, pp. 32–74, 2017.

[19] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security–a survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, Dec 2017.

[20] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978. [Online]. Available: http://doi.acm.org/10.1145/359545.359563

[21] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, pp. 299–319, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/98163.98167

[22] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939 – 948, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2351978917304067

[23] C. Gehrmann and M. A. Abdelraheem, "Iot protection through device to cloud synchronization," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2016, pp. 527–532.

[24] G. N. Schroeder, C. Steinmetz, C. E. Pereira, and D. B. Espindola, "Digital twin data modeling with automationml and a communication methodology for data exchange," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 12 – 17, 2016, 4th IFAC Symposium on Telematics Applications TA 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896316325538

[25] J. Smith and R. Nair, *Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[26] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "Oblivm: A programming framework for secure computation," in *2015 IEEE Symposium on Security and Privacy*, May 2015, pp. 359–376.

[27] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: Trustworthy data analytics in the cloud using sgx," in *2015 IEEE Symposium on Security and Privacy*, May 2015, pp. 38–54.

[28] N. Paladi, C. Gehrmann, and A. Michalas, "Providing user security guarantees in public infrastructure clouds," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 405–419, July 2017.

[29] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 973–990. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[30] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 2019 IEEE Symposium on Security and Privacy (SP)*, vol. 00, 2019, pp. 19–37. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2019.00002

[31] D. Dolev and A. C. Yao, "On the security of public key protocols," in *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, ser. SFCS '81. Washington, DC, USA: IEEE Computer Society, 1981, pp. 350–357. [Online]. Available: https://doi.org/10.1109/SFCS.1981.32

[32] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2014.

[33] S. Kent and K. Seo, "Security architecture for the internet protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4301

[34] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[35] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6347.txt

[36] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful intrusion detection for high-speed network's," in *Proceedings 2002 IEEE Symposium on Security and Privacy*, May 2002, pp. 285–293.

[37] "extensible access control markup language (xacml) version 3.0," OASIS Standard, 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[38] B. Campbell, C. Mortimore, and M. Jones, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants," RFC 7522, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7522.txt

[39] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6749.txt

[40] L. E. Li and T. Woo, "Vsite: A scalable and secure architecture for seamless l2 enterprise extension in the cloud," in *2010 6th IEEE Workshop on Secure Network Protocols*, Oct 2010, pp. 31–36.

[41] T. R. Alves, M. Buratto, F. M. de Souza, and T. V. Rodrigues, "Openplc: An open source alternative to automation," in *IEEE Global Humanitarian Technology Conference (GHTC 2014)*, Oct 2014, pp. 585–589.

Christian Gehrmann received the M.Sc. degree in electronic engineering and the Ph.D. degree in information theory from Lund University, Lund, Sweden, in 1991 and 1997, respectively.

He is an adjunct professor in computer security at Lund University and leads two major research project devoted to security in next generation production systems. His main research interest is in secure systems design, secure execution environments and security protocols. Prof. Gehrmann has been active in many industry standardization bodies and made major contributions to the Bluetooth, Trusted Computing Group and ONVIF (network video) standards. He has been active in research and development of secure computer and communication systems for more than 25 years. He has numerous scientific publications and patents in the information security area that received more than 3400 citations and with and H-index equal to 32.

Martin Gunnarsson is a PhD student at Lund University, Lund, Sweden. His current research focus is security in industrial control systems. He received his Master of Science degree in Computer Science from Lund University in 2017. He has been working in the Security Lab of RISE since 2017 with a focus on communications security for constrained devices. He has been involved in IoT security standardization work at IETF.