R-Chain: A Universally Composable Relay Resilience Framework for Smart Grids

Abubakar Sadiq Sani*‡, Dong Yuan†, Ke Meng*, and Zhao Yang Dong*
* School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia
Email: [sadiq.sani, ke.meng, joe.dong]@unsw.edu.au
† School of Electrical and Information Engineering, The University of Sydney, Sydney, Australia
Email: dong.yuan@sydney.edu.au
‡ School of Computing and Mathematical Sciences, University of Greenwich, London, United Kingdom
Email: s.sani@greenwich.ac.uk

***Abstract*—Smart grids can be exposed to relay attacks (or wormhole attacks) resulting from weaknesses in cryptographic operations such as authentication and key derivation associated with process automation protocols. Relay attacks refer to attacks in which authentication is evaded without needing to attack the smart grid itself. By using a universal composability model that provides a strong security notion for designing cryptographic operations, we formulate the necessary relay resilience settings for strengthening authentication and key derivation and enhancing relay security in process automation protocols in this paper. We introduce R-Chain, a universally composable relay resilience framework that prevents bypass of cryptographic operations. Our framework provides an ideal chaining functionality that integrates all cryptographic operations such that all outputs from a preceding operation are used as input to the subsequent operation to support relay resilience. We apply R-Chain to provide relay resilience in a practical smart grid process automation protocol, namely WirelessHART.**

***Index Terms*—relay resilience, universal composability, authentication, key derivation, smart grids.**

I. Introduction

Smart grids are susceptible to relay attacks by which an active adversary initiates communication between two devices or users to bypass security defences or recover secret cryptographic keys (see, e.g., [1]). Some process automation protocols such as WirelessHART [2] for smart grid communication have been shown to be vulnerable to relay attacks because of weak authentication and/or cryptographic keys. While stronger key derivation schemes and authentication protocols can be applied to mitigate relay attacks, many of such schemes and protocols have vulnerabilities such as lack of randomization, inability to verify and validate the computation time of cryptographic operations, and use of pre-established shared secret keys. This implies that smart grids rely on insecure schemes and protocols. An adversary in smart grid, for example, can use a genuine user as a relay to propagate relay attacks [1]. Note that the terms "user" and "device" are often used interchangeably in this paper.

In the wake of relay vulnerability disclosures in several process automation protocols such as WirelessHART, Modbus [3] and Foundation Fieldbus [4], the community reacted by proposing or recommending key derivation schemes or authentication protocols [1], [5]. Furthermore, Distributed Network Protocol 3 (DNP3) [6] used in smart grids is supported by an authentication protocol during process automation. However, our research shows that when an authentication protocol is not properly integrated with a key derivation scheme, relay attacks can still be carried out, for example, by deriving shared secret keys between unauthenticated users. Thus, the lack of integrating the key derivation scheme and authentication protocol opens opportunities for relay attacks. Furthermore, such integration must be carefully designed to prevent the bypassing of cryptographic operations that support relay resilience.

In this paper, we propose a relay resilience framework, i.e., R-Chain, for providing a relay resilience solution using integrated cryptographic operations, knowledge of computation time of the operations, and cryptographic timestamps based on universal composability [7], which allows the modular design of complex cryptographic protocols. More specifically, our contributions are as follows: (I) We extend the Kusters's theorem that handles concurrent composition of a fixed number of protocol systems to handle the chaining of cryptographic operations. This is a crucial step as many protocols in smart grids are not able to not ensure relay resilience because of cryptographic operations bypass. (II) We present an ideal chaining functionality F_{CC} that uses our new theorem and supports several cryptographic primitives. (III) We propose and prove a realization P_{CC} of F_{CC}. (V) We implement R-Chain. (V) We describe and mitigate relay vulnerabilities in the WirelessHART protocol.

II. Preliminaries

A. The General Notion of Universal Composability

In universal composability models, we have real and ideal protocols. An ideal protocol, also known as ideal functionality, represents the desired behaviour and intended security properties of a protocol. The real protocol, also known as real functionality, represents the protocol to design and analyze and should be at least as secure as the ideal protocol, i.e., it realizes the ideal protocol. For the ideal protocol, there exists an ideal adversary or a simulator while a real adversary exists

for the real protocol, such that there is no environment that can distinguish between real and ideal settings.

B. Inexhaustible Interactive Turing Machine (IITM) Model

The IITM model [7] with responsive environments from [8] is a universal composability model that consists of a general computational model and provides several composition theorems. The general computational model is defined by systems of interactive Turing machines. There are three different types of systems in the IITM model: i) real and ideal protocols/functionalities; ii) adversaries and simulators; and iii) environments. The real and ideal protocols and the environmental systems have an Input/Output (I/O) and network interfaces or tapes, while the adversarial systems have only network tapes. We say that the environmental and adversarial systems are responsive if they immediately respond to so-called *restricting messages* on the network. Restricting messages are represented in the form $(\text{Respond}, id, m)$, where id and m are random bit strings. They are used for enforcing the natural execution of a protocol in universal composability.

C. Kusters's Composition Theorem

Kusters's composition theorem of a fixed number of protocol systems is one of the theorems provided by the IITM model. We first recall the definition of simulation-based security in universal composability before presenting the theorem.

Definition 1 (Strong simulatability) [7]. *Let P and F be protocol systems with similar I/O interface, i.e., the real and ideal protocol, respectively. Then, P realizes F ($P \leq F$) if there exists an adversarial system S (an ideal adversary or a simulator) such that P and $S|F$ have similar external interfaces and for all environmental systems E, connecting to the external interface of P (and thus, $S|F$), it holds true that $E|P \equiv E|S|F$.*

The Kusters's composition theorem handles the concurrent composition of a fixed number of protocol systems.

Theorem 1 [7]. *Let $P_1, ..., P_y, F_1, ..., F_y$ be protocol systems such that $P_1, ..., P_y$ and $F_1, ..., F_y$ only connect with each other via their I/O interfaces and $P_j \leq F_j$, for $j \in \{1, ..., y\}$. Then, $P_1|...|P_y \leq F_1|...|F_y$.*

III. Extended Composition Theorem and Notion of Relay Resilience

A. Extended Notions of Simulation-Based Security

We extend the definition of strong simulatability. To do this, we equip the real and ideal protocols/functionalities with a sequence of cryptographic operations such that an operation always depends on the preceding one, i.e., the operation relies on the preceding operation, to prevent bypass of cryptographic operations and support relay resilience. A sequence of operations c is of the form $c = \{c_1, c_2, ..., c_{n-1}, c_n\}$, where c_1, c_2, c_{n-1}, and c_n are cryptographic operations and c_1 is the first operation of c and preceding operation of c_2 and c_n is the last operation of c. In c, a cryptographic operation, say c_i, can be executed by using all outputs from c_{i-1} as input. In this case, we say that c_i relies on c_{i-1}, i.e., $c_i(c_{i-1})$.

Definition 2 (Extended strong simulatability). *Let P and F be a real protocol and an ideal protocol, respectively, with the same I/O interface. Let c be a sequence of cryptographic operations of P and F. Then, $P[c] \leq_R F[c]$ iff every operation c_i in c always rely on the preceding operation c_{i-1}, and there exists an adversarial system S such that the systems $P[c]$ and $S|F[c]$ have the same external interface and for all environmental systems E, connecting only to the external interface of $P[c]$ and $S|F[c]$, it holds true that $E|P[c] \equiv E|S|F[c]$, where the adversary in $E|P[c]$ is subsumed by E.*

B. Extended Composition Theorem

Our extended theorem handles concurrent composition of a fixed number of protocol systems with a sequence of cryptographic operations that rely on one another.

Theorem 2. *Let P_1, P_2, F_1, F_2 be protocol systems with a sequence c of cryptographic operations c_1, c_2, and c_3 such that P_1 and P_2 as well as F_1 and F_2 only connect with each other via their I/O interfaces and for every k, $P_k[c] \leq_R F_k[c]$, iff $P_k[c_i(c_{i-1})] \leq_R F_k[c_i(c_{i-1})]$, where $i \in \{1, 2, 3\}$. Then, $P_1[c_3(c_2(c_1))]|P_2[c_3(c_2(c_1))] \leq_R F_1[c_3(c_2(c_1))]|F_2[c_3(c_2(c_1))]$, for $k \in 1, 2$.*

Proof Sketch: We prove the theorem for $k = 2$ and $i = \{1, 2, 3\}$. Let $S = S_1|S_2$. Since $P_1[c] \leq_R F_1[c]$ and $P_2[c] \leq_R F_2[c]$, we have $E|P_1[c_3(c_2(c_1))] \equiv E|S_1|F_1[c_3(c_2(c_1))]$, and $E|P_2[c_3(c_2(c_1))] \equiv E|S_2|F_2[c_3(c_2(c_1))]$. Based on our definition of extended notions of simulation-based security, we now obtain

$$E|P_2[c_3(c_2(c_1))]|P_1[c_3(c_2(c_1))] \equiv E|S_2|F_2[c_3(c_2(c_1))]|S_1|F_1[c_3(c_2(c_1))] \quad \text{(Definition of } S\text{)}$$

C. The Notion of Relay Resilience

Relay resilience is motivated by cryptographic protocols that despite negotiating an output remain vulnerable to relay attacks. Relay security complements the correctness of negotiation: A protocol is relay secure when two users always negotiate a shared secret session key based on their attributes and knowledge of the computation time of cryptographic operations and a cryptographic timestamp, which securely proves that a cryptographic proof was computed at a specific time. Hence, relay security concerns conditions in which one device can protect the other device, even if the latter supports insecure usage of cryptography. However, we have to assume that some of the mechanisms of the protocol, e.g., encryption and Pseudo-Random Function (PRF), are strong. On the other hand, there is no cryptographically approach to prevent an adversary from carrying out a relay attack if both users enable a completely insecure negotiation. Our relay resilience definition is parameterized by attributes of users, a sequence of cryptographic operations, and cryptographic timestamp from which we expect relay protection.

Definition 3 (Relay resilience). *Let se be a session between users ID_A and ID_B with a set of attributes att_A and att_B, respectively. Let c be a sequence of cryptographic operations in se using att_A and att_B to establish a cryptographic timestamp $ctimestamp$. Then, se is relay resilient iff $se(ctimestamp)$ is valid for $se(att_A.att_B.c)$.*

A relay attack means that a session is weaker than the prescribed one. An agreement on the users' attributes and knowledge of the cryptographic operations and cryptographic timestamps is essential for relay protection. We have relay protection if the agreement succeeds. Note that only sessions with two users get relay protection guarantees. The users' attributes for relay protection play a role similar to authentication while the knowledge of the computation time of cryptographic operations and cryptographic timestamps play a key derivation role. These show that relay protection should depend on inputs to the negotiation and the negotiation itself.

IV. R-Chain

In this section, we present R-Chain, which provides an ideal chaining functionality F_{CC} that supports cryptographic primitives in smart grids. A smart grid process automation protocol P can use F_{CC} for its cryptographic operations c. Then, we can show that P using $F_{CC}[c]$ realizes some ideal key derivation functionality Fk, i.e., $P|F_{CC}[c] \leq_R Fk$. Once $P|F_{CC}[c] \leq_R Fk$ is proven using the extended composition theorem (cf. Section III), F_{CC} can be replaced with its realization P_{CC} (see below), where all the ideal operations provided by F_{CC} are replaced by their corresponding real operations. Formally, F_{CC} is a machine with n I/O tapes and a network tape. The I/O tapes represent different roles in a process automation protocol while the network tape is used for communicating with the adversary. In every run of a system which uses F_{CC} for integrated cryptographic operations, only one instance of F_{CC} will always be available to handle all requests for relay resilience support. A user of F_{CC} is identified by a tuple (ID, sid, r), where ID is the user identity, sid is a session identifier, and r is the role/tape which connects the user to F_{CC}, ID, and sid, which is chosen and managed by process automation protocols. (ID, sid) are used for prefixing all messages on the I/O tapes so that every user that sent/receives a message can be identified by F_{CC}.

We parameterize F_{CC} with an ECDH domain parameters algorithm $EDPGen(1^\eta)$ and a timestamp function f that takes a message m and outputs a timestamp $f(m) = t$, where η is a security parameter. $EDPGen(1^\eta)$ takes η as input and efficiently computes and returns (p, a, b, G, n, h), where p is a prime modulus, a and b are curve parameters, G is a general point, n is the order of G, and h is a co-factor. Private keys and other secret keys derived from the private keys in F_{CC} are modelled in such a way that users do not get the actual secret keys but rather get pointers to such keys to provide extra protection to the keys (note that before a message is used with a pointer, the pointer is replaced by the key it refers to). F_{CC} maintains the actual values of all secret keys and the pointers that point to these keys, ephemeral

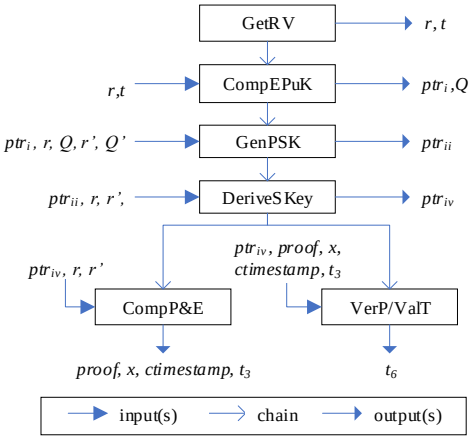

Fig. 1. A Simple Description of R-Chain.

random values, and cryptographic timestamps in a distributed database that is created by a Smart Grid Authority (SGA). We assume that F_{CC} accurately synchronizes its clock with that of the SGA's distributed database for timestamp accuracy. F_{CC} executes $EDPGen(1^\eta)$ upon its first activation and store the generated (p, a, b, G, n, h). Furthermore, we expect F_{CC} to receive users' identities (Users, IDs). Note that the received information can be made public. The cryptographic commands that F_{CC} provides to a key derivation initiator (ID, sid, r) and a key derivation responder (ID', sid', r') on their respective I/O interfaces are provided in Table I.

In R-Chain, we construct a realization P_{CC} of F_{CC} by using standard cryptographic schemes to implement all F_{CC}'s commands. Formally, P_{CC} is a machine that has the same I/O and network interfaces as F_{CC}. It is parameterized with the $EDPGen(1^\eta)$ algorithm and a timestamp function f with similar properties as F_{CC}, an encryption scheme $Enc(.)/Dec(.)$, a message authentication code $MAC(.)/VMAC(.)$, a hash function $H(.)$, and two families of PRF F and F' that take key(s) and salt as input and output a key. A detailed description of how each of the commands in F_{CC} is implemented in P_{CC} is provided in Table II. Furthermore, a simple description of R-Chain is depicted in Fig. 1, which shows that every output from a preceding operation is used as input in the subsequent operation. One could easily see that if the right data is not provided to execute a command, such command cannot be executed.

V. R-Chain Implementation

In this section, we implement R-Chain on low power wireless sensors, i.e., Tmote Sky mote sensors, which are widely used in smart grids, and further simulate it using the Network Simulator 3 (NS-3) tool to show that it meets our 20 $msec$ latency target of smart grid applications such as distributed energy resources. The details of the simulation parameters we used in NS-3 are as follows: i) Platform is Ubuntu 16.04 LTS; ii) Communication medium is Wi-Fi; iii) Transport layer is UDP; and iv) Communication range 100 $metres$. We also consider other standard NS-3 parameters such as measuring network protocols performance

TABLE I
CRYPTOGRAPHIC COMMANDS OF THE IDEAL CHAINING FUNCTIONALITY F_{CC}

Cryptographic Commands
Generate a fresh ephemeral random value [(GetRV)]. The user (ID, sid, r) can request F_{CC} to generate a fresh ephemeral random value r. Upon receiving this request, F_{CC} forwards the request to the adversary via a restricting message. The adversary is supposed to provide $r \in \{1, ..., nr\}$ at a timestamp t, where nr is a large randomly selected integer. F_{CC} checks whether r is fresh to prevent ephemeral random value collision. If r already exists in the database, F_{CC} requests another r until the check succeeds. Then, F_{CC} adds (ID, r, t) to the database and returns (RV, r, t) to the user. Note that the user $(ID^{'}, sid^{'}, r^{'})$ can also execute this command.
Generate a fresh ephemeral private and compute its corresponding ephemeral public key [(CompEPuK, r, t)]. The user (ID, sid, r) can request F_{CC} to generate a fresh ephemeral private d and compute its corresponding ephemeral public key Q. Upon receiving this request, F_{CC} checks whether (ID, r, t) exists in the database and forwards this request to the adversary via a restricting message if this check succeeds. The adversary is supposed to provide $d \in \{1, ..., n\}$. F_{CC} ensures that d is fresh to prevent a private key collision. If d is fresh, F_{CC} stores a pointer ptr_i that points to d for the user, uses (p, a, b, G, n, h) to compute a public key $Q = d.G$, adds (r, Q) and d to the database, and returns (EPuK, ptr_i, Q) to the user. Note that the user $(ID^{'}, sid^{'}, r^{'})$ can also execute this command.
Generate a fresh preshared key [(GenPSK, $ptr_i, r, Q, r^{'}, Q^{'}$)]. The user (ID, sid, r) can request F_{CC} to generate a fresh preshared key. Upon receiving this request, F_{CC} checks whether (r, Q) and $(r^{'}, Q^{'})$ are recorded in the database. If the checks succeed, F_{CC} further checks whether a preshared key k has been generated by $(r, Q, r^{'}, Q^{'})$ and returns a pointer ptr_{ii} to k if the check succeeds. Otherwise, F_{CC} generates a new preshared key as follows. F_{CC} forwards the request to the adversary (via a restricting message to provide a new preshared key $k = F_\eta(H((d.Q^{'}), (r^{'}, r)))$ using SHA-256 algorithm $H(.)$ and PRF F, where $d.Q^{'}$ represents an ECDH key. F_{CC} ensures that k is fresh and has been generated as $F_\eta(H((d.Q^{'}), (r^{'}, r)))$, adds k and $(r, r^{'})$ to the database, sets the pointer ptr_{ii} to k, and returns $(\text{PSKPointer}, ptr_{ii})$ to the user. Note that the user $(ID^{'}, sid^{'}, r^{'})$ can also execute this command.
Shared secret session key derivation [(DeriveSKey, $ptr_{ii}, r, r^{'}$)]. The user (ID, sid, r) can request F_{CC} to derive a shared secret session key k_i. Upon receiving this request, F_{CC} first checks whether $(r, r^{'})$ exists in the database. If the check succeeds, F_{CC} checks whether k_i has been derived using pointer ptr_{ii}, r, and $r^{'}$, and outputs the pointer ptr_{iii} pointing to k_i. Otherwise, F_{CC} forwards this request to the adversary via a restricting message to compute k_i using $H(.)$. F_{CC} ensures that k_i is fresh and it is derived as $k_i = F_\eta^{'}(H(k.r.r^{'}))$. Then, F_{CC} adds k_i to the database, stores a new pointer ptr_{iv} pointing to k_i for the owners of k and returns $(\text{SKeyPointer}, ptr_{iv})$ to the user. Note that the user $(ID^{'}, sid^{'}, r^{'})$ can also execute this command.
Compute a cryptographic proof and a cryptographic timestamp [(CompP&E, $ptr_{iv}, r, r^{'}$)]. This command is provided to only the user (ID, sid, r). Upon receiving this request to compute a cryptographic proof and a cryptographic timestamp, F_{CC} checks whether ptr_{iv} belongs to the user and $(r, r^{'})$ exists in the database. If these checks succeed, it provides a cryptographic proof $proof$ and a cryptographic timestamp $ctimestamp$ using an AES 128-bit encryption algorithm $Enc(.)$ and 256-bit MAC algorithm $MAC(.)$. F_{CC} computes $proof = MAC_{k_i}(x)$ at a timestamp t_3, where $x = Enc_{k_i}(k_i, r, r^{'})$ and $ctimestamp = t_3.H(k_i, r, r^{'})$ for the user. Then, F_{CC} stores $(proof, x, ctimestamp, t_3)$ for k_i in the database, and returns $(\text{P\&E}, proof, x, ctimestamp, t_3)$ to the user.
Verify a cryptographic proof and validate a cryptographic timestamp [(VerP/ValT, $ptr_{iv}, proof, x, ctimestamp, t_3$)]. This command is provided to only the user $(ID^{'}, sid^{'}, r^{'})$. Upon receiving this request to verify a cryptographic proof and validate a cryptographic timestamp, F_{CC} first checks whether ptr_{iv} belongs to the user and then uses a 256-bit MAC verification algorithm $VMAC(.)$ and decryption of the AES 128-bit encryption algorithm $Dec(.)$ to verify $proof$ as follows: i) $VMAC_{k_i}(proof, x) = 1$? at a timestamp t_6; and ii) $(k_i, r, r^{'}) = Dec_{k_i}(x)$. If the verifications succeed, F_{CC} validates $ctimestamp$ as follows: i) computes $t_3.H(k_i, r, r^{'}) = ctimestamp$; and ii) checks whether there exists exactly $ctimestamp$ such that $ctimestamp$ is stored for k_i. If the validations succeed, F_{CC} returns $(\text{Validation}, t_6)$ to the user. Otherwise, F_{CC} returns $(\text{Validation}, restricted)$ to the user.

TABLE II
IMPLEMENTATION OF F_{CC} COMMANDS IN ITS REALIZATION P_{CC}

Implementation of Cryptographic Commands
Generate a fresh ephemeral random value [(GetRV)]. P_{CC} selects $r \leftarrow \{1, ..., nr\}$ at timestamp t and outputs (r, t) to the user.
Generate a fresh ephemeral private and compute its corresponding ephemeral public key [(CompEPuK, r, t)]. P_{CC} checks whether ID, r, and t are valid, selects d, creates a pointer ptr_i to d, uses the domain parameters to compute $Q = d.G$, and outputs (ptr_i, Q) to the user if the checks succeed.
Generate a fresh preshared key [(GenPSK, $ptr_i, r, Q, r^{'}, Q^{'}$)]. P_{CC} checks whether (r, Q) and $(r^{'}, Q^{'})$ are valid and returns $(\text{PSKPointer}, restricted)$ to the user if any of these checks fails. Otherwise, P_{CC} computes $k = F_\eta(H((d.Q^{'}), (r^{'}, r)))$, where d is the private key of the user to which the pointer ptr points to, creates a new pointer ptr_{ii} to k, and returns ptr_{ii} to the user.
Shared secret session key derivation [(DeriveSKey, $ptr_{ii}, r, r^{'}$)]. P_{CC} checks whether r and $r^{'}$ are valid, computes $k_i = F_\eta^{'}(H(k.r.r^{'}))$, creates a pointer ptr_{iv} to k_i, and returns ptr_{iv} to the user if the checks succeed.
Compute a cryptographic proof and a cryptographic timestamp [(CompP&E, $ptr_{iv}, r, r^{'}$)]. P_{CC} checks whether ptr_{iv} is recorded for ID and r and $r^{'}$ are valid, computes $x = Enc_{k_i}(k_i, r, r^{'})$, $proof = MAC_{k_i}(x)$ at a timestamp t_3, and $ctimestamp = t_3.H(k, r, r^{'})$ and then returns $(proof, x, ctimestamp, t_3)$ to the user if the checks succeed.
Verify a cryptographic proof and validate a cryptographic timestamp [(VerP/ValT, $ptr_{iv}, proof, x, ctimestamp, t_3$)]. P_{CC} verifies whether ptr_{iv} is recorded for ID and $VMAC(proof, x) = 1$ and $(k_i, r, r^{'}) = Dec_{k_i}(x)$. If the verifications succeed, P_{CC} validates that $t_3.H(k_i, r, r^{'}) = ctimestamp$ and then returns t_6 to the user if the validation succeeds.

using flow monitor. In R-Chain, the computation time of AES 128-bit encryption (T_{se}), AES 128-bit decryption (T_{sd}), SHA-256 (T_{ha}), 160-bit point multiplication (T_{pm}), 256-bit MAC/VMAC (T_{hm}), and 32-bit random number (T_{rn}) is 0.0017 sec, 0.00167 sec, 0.0091 sec, 1.04 sec, 0.0183 sec, and 0.0073 $msec$, respectively. We assess the maximum computational cost required for R-Chain execution and our results show that the computational cost required for users ID (executing the commands GetRV, CompEPuK, GenPSK, DeriveSKey, and CompP&E) and $ID^{'}$ (executing the commands GetRV, CompEPuK, GenPSK, DeriveSKey, and VerP/ValT) is $4T_{rn} + 2T_{pm} + 4T_{ha} + T_{hm} + T_{se} \approx 2.13643$ sec and $4T_{rn} + 2T_{pm} + 4T_{ha} + T_{hm} + T_{sd} \approx 2.13639$ sec, respectively. The End-to-End Delay (EED) of R-Chain is ≈ 6.5146 $msec$. We can see that the EED is less than our 20 $msec$ latency target. Thus, R-Chain is fit for the smart grids. Due to page limit, more details on R-Chain implementation and additional performance evaluation will be provided in our future work.

VI. Case Study

In this section, we carry out a case study to show the usefulness of our framework by analysing and enhancing the WirelessHART protocol [2], which is a communication protocol designed for industrial process automation and control and is meant to provide relay resilience in smart grids. The essential entities in the WirelessHART protocol network include network manager, security manager, gateway, field devices and handheld devices. Security keys in the WirelessHART protocol are managed and distributed by the network manager in collaboration with the security manager. For a secure communication session between two devices, the devices request for a session key, which is used for securing and maintaining the session. The network manager can broadcast messages to all the field devices using a similar key. Hence, we deduce that the WirelessHART protocol also relies on a pre-established secret key for secure communication. A simple description of the WirelessHART protocol is depicted in Fig. 2. A malicious device, say ID_B, can perform relay attacks in WirelessHART protocol by using its knowledge of the session key and messages sent by the network manager to relay the messages to an arbitrary device, say ID_C. In this case, ID_B relays the messages to ID_C in the network. As a result, the message will be received and executed by ID_C. To see this, we consider a setting of three machines $M_{A(WH)}$, $M_{B(WH)}$, and $M_{C(WH)}$ to model the roles of the initiator, responder, and arbitrary node, respectively. An honest initiator instance outputs an encrypted message m generated using a pre-established secret key k that is known to an honest responder instance and an arbitrary node. The responder instance that received m can forward it to the arbitrary node. The arbitrary node instance might have received m, which was relayed by the responder. In this case, the protocol does not guarantee that m is a relayed message as k is known by the instances. Thus, we have no security guarantee for k and m, and the responder can easily let the arbitrary node instance accept m. While this form of relay attack is not a direct attack against the protocol, this setting shows that security of the protocol is not sufficient to mitigate relay attack. To fix this problem in our setting, we enhance the protocol as follows: i) establish a preshared key between the responder and arbitrary node using GetRV, CompEPuK, and GenPSK commands of F_{CC} to avoid reliance on a pre-established shared secret key and provide a security guarantee for the preshared key; ii) equip the protocol with a session key derivation via the DeriveSKey command of F_{CC} for every communication session between the responder and arbitrary node to provide a security guarantee for m and communication session and support relay resilience; and iii) equip the protocol with CompP&E and VerP/ValT commands of F_{CC} to provide relay resilience.

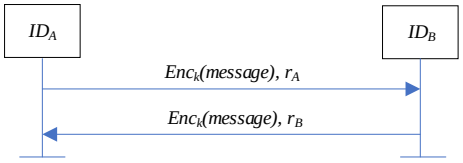

Fig. 2. WirelessHART protocol. Abbreviations: k – pre-established shared secret key, r_A and r_B – random values.

VII. Conclusion

In this paper, we have proposed R-Chain, a universally composable relay resilience framework for analyzing relay security and providing relay resilience in smart grid process automation protocols. R-Chain extended the Kusters's universal composition theorem on a fixed number of protocol systems and consists of an ideal chaining functionality that provides integrated cryptographic operations for relay resilience. We have demonstrated the usefulness of R-Chain in a case study, namely the WirelessHART protocol. We uncovered some weaknesses in the relay security of the protocol and used R-Chain to provide relay resilience by enhancing the security of the protocols. In future work, we will apply R-Chain to other smart grid process automation protocols, provide its detailed performance evaluation, and extend it to further mitigate ransomware attacks against the smart grids.

References

[1] L. Bayou, D. Espes, N. Cuppens-Boulahia, and F. Cuppens, "Security analysis of wirelesshart communication scheme," in *9th International Symposium on Foundations and Practice of Security*. Springer, 2016, Conference Proceedings, pp. 223–238.

[2] F. Group, 2018. [Online]. Available: https://fieldcommgroup.org/

[3] Modbus, "Modbus protocol specification," 2018. [Online]. Available: http://www.modbus.org/specs.php

[4] F. Group, "Foundation fieldbus," 2020. [Online]. Available: https://fieldcommgroup.org/technologies/foundation-fieldbus

[5] K. Imtiaz and M. J. Arshad, "Security challenges of industrial communication protocols: Threats vulnerabilities and solutions," *International Journal of Computer Science and Telecommunications*, 2019.

[6] D. U. Group, "Overview of DNP3 protocol," 2020. [Online]. Available: https://www.dnp.org/About/Overview-of-DNP3-Protocol

[7] R. Kusters, "Simulation-based security with inexhaustible interactive turing machines," in *19th IEEE Computer Security Foundations Workshop (CSFW'06)*. IEEE, 2006, pp. 12–pp.

[8] J. Camenisch, R. R. Enderlein, S. Krenn, R. Küsters, and D. Rausch, "Universal composition with responsive environments," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2016, pp. 807–840.