Crypto-Chain: A Relay Resilience Framework for Smart Vehicles

Abubakar Sadiq Sani
University of Greenwich
London, United Kingdom
s.sani@greenwich.ac.uk

Dong Yuan
The University of Sydney
Sydney, Australia
dong.yuan@sydney.edu.au

Elisa Bertino
Purdue University
West Lafayette, Indiana, USA
bertino@purdue.edu

Zhao Yang Dong
University of New South Wales
Sydney, Australia
joe.dong@unsw.edu.au

ABSTRACT

Recent findings show that smart vehicles can be exposed to relay attacks resulting from weaknesses in cryptographic operations, such as authentication and key derivation, or poor implementation of these operations. Relay attacks refer to attacks in which authentication is evaded without needing to attack a smart vehicle itself. They are a recurrent problem in practice. In this paper, we formulate the necessary relay resilience settings for strengthening authentication and key derivation and achieving the secure design and efficient implementation of cryptographic protocols based on universal composability, which allows the modular design and analysis of cryptographic protocols. We introduce Crypto-Chain, a relay resilience framework that extends Kusters's universal composition theorem on a fixed number of protocol systems to prevent bypass of cryptographic operations and avoid implementation errors. Our framework provides an ideal crypto-chain functionality that supports several cryptographic primitives. Furthermore, we provide an ideal functionality for mutual authentication and key derivation in Crypto-Chain by which cryptographic protocols can use cryptographic operations, knowledge about the computation time of the operations, and cryptographic timestamps to ensure relay resilience. As a proof of concept, we first propose and implement a mutual authentication and key derivation protocol (*MKD*) that confirms the efficiency and relay resilience capabilities of Crypto-Chain and then apply Crypto-Chain to fix two protocols used in smart vehicles, namely Megamos Crypto and Hitag-AES/Pro.

KEYWORDS

relay resilience, universal composability, smart vehicles, key exchange, mutual authentication and key derivation

1 INTRODUCTION

Smart vehicles are susceptible to relay attacks [14] by which an active adversary initiates communication between two devices or users to bypass security defences or recover secret cryptographic keys. A number of cryptographic protocols such as Megamos Crypto [5, 36] used in smart vehicles have been shown to be vulnerable to relay attacks (see, e.g., [32]) because of weak authentication and/or cryptographic keys. Stronger key derivation schemes and authentication protocols have thus been proposed for mitigating such attacks [25, 35, 39]. However, such schemes and protocols have vulnerabilities such as lack of key update and randomization, use of pre-established shared secret keys, and use of unreliable timing information via the inability to verify and validate the computation time of cryptographic operations. These imply that smart vehicles rely on insecure cryptographic protocols. An adversary can bypass the security defences of a car to open the door locks of the car [36]. Note that in what follow we use the term "user" to refer to a smart vehicle device like a smart sensor and therefore the "user" and "device" terms are often used interchangeably.

Cryptographic protocols have proven important in mitigating relay attacks in smart vehicles. For example, in the wake of relay vulnerability disclosures in the Megamos Crypto, the community reacted by either proposing key derivation schemes or authentication protocols [34], [12]. 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 intertwining key derivation scheme and authentication protocols opens opportunities for relay attacks. However, such integration must be carefully designed to avoid relying on weak cryptographic primitives and prevent the bypassing of cryptographic operations that support relay resilience.

Surprisingly, there has been very little formal workaround intertwining key derivation schemes and authentication protocols (with the support of the computation time of cryptographic operations and cryptographic timestamps) in modern relay resilience solutions for smart vehicles. Recent approaches either provide key derivation schemes (based on pre-stored passwords or pre-established shared secret keys) or authentication protocols for relay resilience [25, 38] but not both. Other work has focused on the implementation of attacks and provided some countermeasures such as access control restrictions and generation of secret keys [14, 36]. All such approaches neither guarantee relay resilience nor can protect from certain relay attacks that can occur when an unauthenticated user derives a key. This is a major issue because lack of authentication is fertile ground for relay attacks. Recent relay attacks have shown the exploitation of unauthenticated users supporting unauthenticated key derivation or negotiation. In this work, we aim to address these issues by investigating the problem of relay resilience in cryptographic protocols of smart vehicles and providing a novel relay resilience solution using integrated cryptographic operations, knowledge of computation time of the operations, and cryptographic timestamps based on universal composability [18, 22], which allows the modular design of complex cryptographic protocols (see Section 4.1 for a description of cryptographic timestamps).

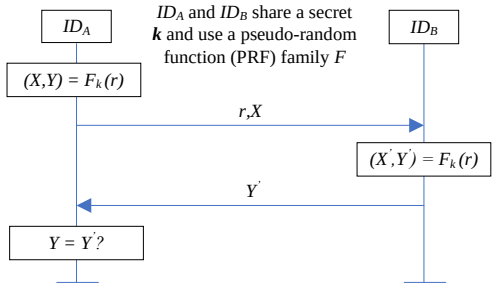

Figure 1: Megamos Crypto authentication protocol.

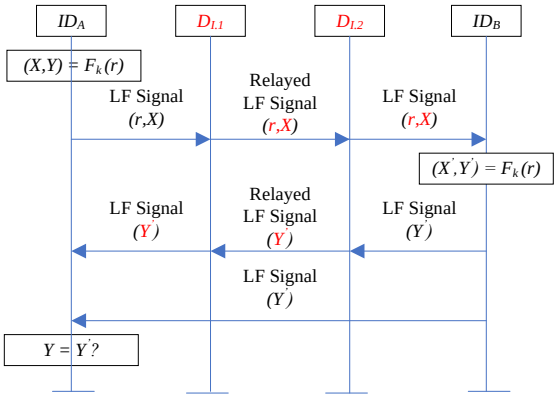

Figure 2: Relay Attacks on Megamos Crypto authentication protocol.

1.1 Motivating Example

We start with a motivating example. Consider the Megamos Crypto authentication protocol [34] in smart cars for negotiating an output from a pseudo-random function (PRF) family for authentication as shown in Figure 1. In the first message, ID_A (say, an electronic vehicle immobilizer) presents a random number and one part of its PRF output; in the second message, ID_B (say, a transponder car key) indicates one part of its PRF output. The goal of this protocol is to compute (X, Y) and (X', Y') and verify that $Y = Y'$ so that ID_B authenticates its identity to ID_A. The Megamos Crypto succeeds under normal circumstances. However, consider a scenario where an adversary I is using relay attack devices $D_{I.1}$ and $D_{I.2}$ (for capturing low frequency (LF) or radio frequency (RF) signals) located between ID_A and ID_B and ID_A sends the first message to ID_B. This leads to a relay attack (see Figure 2), where I can use $D_{I.1}$ to receive the signal from ID_A and transmit it to $D_{I.2}$ and further use $D_{I.2}$ to transmit the signal to ID_B and compromise the connection between ID_A and ID_B.

A few techniques have been adopted by protocol designers to mitigate such attacks. We identified three common approaches based on a review of widely deployed protocols. The first approach, exemplified by Hitag-AES/Pro [31] from NXP, is to use a proprietary Advanced Encryption Standard (AES) 128-bit algorithm during authentication. The second approach relies on a password prestored in the memory of a device and is best exemplified by Atmel's ATA5795C [3]. The third approach relies on a pre-established shared secret key and is best exemplified by the Hitag-AES/Pro. All these approaches have various advantages and disadvantages. For example, Hitag-AES/Pro fails to use PRF during authentication and is thus prone to relay attacks. Inputs of authentication features in Megamos Crypto also lead to relay attack. Besides, ATA5792 relies solely on pre-stored password which leads to eavesdropping of exchanged data. Furthermore, the protocols are not secure in arbitrary adversarial environments, i.e., they lack universal composition properties [6], [19].

1.2 Overview of our approach

We give a definition of relay resilience that models the intuitive and desirable property for cryptographic protocols such as Megamos Crypto: *To mitigate relay attack on a particular protocol, it is sufficient to chain all cryptographic operations and prove the validity of the computation time of cryptographic operations and cryptographic timestamps.* Our work builds on the definition of strong simulatability in the universal composability and composition theorem of Kusters [22, 23] used to model a fixed number of protocol systems in the universal composability model. In a protocol system, users can bypass cryptographic operations and still be able to continue a protocol execution. As a result, cryptographic operations to mitigate relay attacks may not be executed. Our solution is to incorporate relay resilience in the definition to ensure that (i) the user or an adversary cannot force the execution of preferred cryptographic operations and that (ii) a universally composable relay resilience guarantee is established.

We consider mutual multi-factor authentication and key exchange between users. These users each have their own identities and can express knowledge of computation time of cryptographic operations and cryptographic timestamps and show their intent to negotiate a shared secret session key. The factors used for authentication include the identity of a user, ephemeral random value, ephemeral public and private keys, and a combination of these factors. By ephemeral we mean that the values and keys are used for a single protocol execution and discarded after completion of a key derivation and usage. To formally define relay resilience, we introduce relay protection cryptographic primitives, such as dynamic multi-factor authentication (DMA), which operates on a user's identity and attributes (such as ephemeral random values, timestamps, and ephemeral public keys), and knowledge-based key exchange (KKE), which operates on establishing an authenticated preshared key based on Elliptic Curve Diffie-Hellman (ECDH) key exchange and the user's attributes. We expect relay resilience support from the DMA and KKE primitives.

We also introduce a mutual authentication and key derivation functionality that maps two users' identities and attributes to derive a shared secret session key and compute a cryptographic proof/a cryptographic timestamp for negotiating an output and proving the negotiation, respectively (see Section 4.1 for a description of cryptographic proof). Intuitively, our definition says that a protocol is relay secure if two users, starting from their attributes and knowledge of computation time of cryptographic operations satisfying

DMA and *KKE* to a commonly established timestamp, can only negotiate a shared secret session key determined by the functionality, even in the presence of an adversary.

We adopt the following approach to apply our definition to real-world protocols in smart vehicles. Rather than analyzing the entire protocol, we first extract the authentication procedure and/or key derivation procedure which capture the main relay-protection mechanisms of the protocol. Then, we prove that either or both procedures are complete for relay security.

1.3 Summary of our contributions

To the best of our knowledge, Crypto-Chain is the first framework to simultaneously offer dynamic multi-factor mutual authentication and key derivation with integrated cryptographic operations, knowledge of computation time of the operations, and cryptographic timestamps for relay resilience in smart vehicles. Our primary contribution is a novel relay resilience framework, i.e., Crypto-Chain, for smart vehicles. We devise a methodology to analyze the relay security of complex protocols by analyzing only the authentication and/or key derivation procedures of the protocol. More specifically, our contributions are as follows: (I) We extend Kusters's theorem that handles concurrent composition of a fixed number of protocol systems to handle chainings of cryptographic operations. (II) We present an ideal crypto-chain functionality F_{CC} that uses our new theorem and supports several cryptographic primitives, such as *DMA* and *KKE*. (III) We propose and prove a realization P_{CC} of F_{CC}. (IV) We propose an ideal functionality for mutual authentication and key derivation F_{MKD}. (V) We use Crypto-Chain, which consists of F_{CC} and F_{MKD}, to construct the first mutual authentication and key derivation protocol (*MKD*) based on *DMA* and *KKE* for relay resilience in smart vehicles. We further implement and analyse *MKD*. (VI) We describe and mitigate relay vulnerabilities in Megamos Crypto and Hitag-AES/Pro.

1.4 Outline of the paper

In Section 2, we briefly review the IITM model, which represents the universal composability model we use in this paper, and Kusters's composition theorem we extend in this paper. In Section 3, we present our extended composition theorem, formally define relay resilience, and introduce our threat model. In Section 4, we present Crypto-Chain. In Section 5, we present *MKD*. In Section 6, we implement and analyse *MKD*. In Section 7, we present our case studies. In Section 8, we discuss related work. In Section 9, we conclude and give suggestions for future work. Further details are provided in the appendix.

2 UNIVERSAL COMPOSABILITY

In this section, we briefly recall the general notion of universal composability, the IITM model [18, 22] with responsive environments from [6], and the Kusters's composition theorem. In universal composability models, we have real and ideal protocols/functionalities. 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.

The IITM model 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. An interactive Turing machine (or a machine) is defined as a probabilistic polynomial-time Turing machine with named bidirectional tapes. In a system of IITMs of the form $TM = M_1|...|M_k|!M_1^{'}|...|!M_k^{'}$, where M_i and $M_j^{'}$ are machines and ! indicates that an unbounded number of fresh copies of machines may be generated in a run of TM. A machine M_1 can be triggered by another machine $M_1^{'}$ if $M_1^{'}$ receives a message on a tape that connects them. Two systems Y and Z are called indistinguishable, i.e., $Y \equiv Z$, if the difference between the probability that Y outputs 1 on a special tape and the probability that Z outputs 1 is negligible. There are three different types of systems in the IITM model: i) real and ideal protocols/functionalities; ii) adversaries and simulators; iii) and environments. The real and ideal protocols and the environmental systems have 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 (Respond, id, m), where id and m are random bit strings. They are used for enforcing a natural execution of a protocol by preventing the adversary from disrupting or interfering with the protocol execution. In general, restricting messages improves the expressivity of universal composability models.

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) [22]. *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 $E|P \equiv E|S|F$.*

Kusters's composition theorem handles the concurrent composition of a fixed number of protocol systems is presented below.

Theorem 1 [22]. *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, $P_1|...|P_y \leq F_1|...|F_y$.*

More complex systems can be constructed by combining other composition theorems of the IITM model in [22, 23].

3 EXTENDED COMPOSITION THEOREM, NOTION OF RELAY RESILIENCE, AND THREAT MODEL

In this section, we extend the notions of simulation-based security, state and provide proof of our extended composition theorem, and provide a threat model.

3.1 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 (i) c_i relies on c_{i-1}, i.e., $c_i(c_{i-1})$ and (ii) c_{i-1} is excecuted before c_i. To ensure that c_{i-1} is executed first, c_i requires unique outputs from c_{i-1} execution. If the order of the execution between c_{i-1} and c_i is changed, c_i cannot be successfully executed as the inputs required for its execution are incomplete.

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 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.*

3.2 Extended Composition Theorem

Our extended composition theorem handles the 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

$$E|P_2[c_3(c_2(c_1))]|P_1[c_3(c_2(c_1))] \equiv$$
$$E|S_2|F_2[c_3(c_2(c_1))]|S_1|F_1[c_3(c_2(c_1))]$$

$$\equiv E|S|F_1[c_3(c_2(c_1))]|F_2[c_3(c_2(c_1))] \qquad \text{(Definition of } S\text{)}$$

3.3 Notion of Relay Resilience

Relay resilience is motivated by cryptographic protocols such as Megamos Crypto that despite negotiating an output remain vulnerable to attacks. Although we expect the protocol negotiation to be correct and hold unconditionally as its session is complete, design and implementation errors may break it. For instance, PRF does not offer resilience to relay attacks on Megamos Crypto but still support the negotiation of an output. An implementation of Megamos Crypto in which an initiator, say device ID_A, accepts a random number would also fail to mitigate relay attacks. Note that the term "session" refers to communications between two users.

Relay security complements the correctness of a 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 PRF, are strong. Our relay resilience definition is parameterized by attributes of users, sequence of cryptographic operations, and cryptographic timestamp from which we expect relay protection.

Definition 3 (Relay resilience). *Let s 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 s using att_A and att_B to establish a cryptographic timestamp $ctimestamp$. Then, s is relay resilient iff $s(ctimestamp)$ is valid for $s(att_A.att_B.c)$.*

A relay attack means that a session is weaker than the prescribed one. 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, i.e., initiator and responder, get relay protection guarantees. The users' attributes for relay protection play a role similar to mutual 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.

3.4 Threat Model

In this paper, we use the Dolev-Yao attacker model [13] as our threat model. In the Dolev-Yao attacker model, an adversary may be an outsider or a legitimate device. The adversary can eavesdrop, forge, and relay messages. The adversary in the real world cannot guess and calculate secret values, secret keys, and pointers to the secret keys.

4 CRYPTO-CHAIN

In this section, we present our relay resilience framework, Crypto-Chain, which consists of F_{CC} and F_{MKD} as mentioned in the introduction. In Crypto-Chain, F_{CC} is used for cryptographic operations while F_{MKD} is used for mutual authentication and key derivation between two users in smart vehicles.

4.1 Ideal Crypto-chain Functionality for Cryptographic Primitives

Our ideal crypto-chain functionality F_{CC} supports cryptographic primitives such as our *DMA* and *KKE* for enabling relay resilience in smart vehicles. *DMA* is similar to authentication, except that it

provides dynamic multi-factor authentication, where authentication is performed on a per-session basis such that only the identity of a user always remains the same and other features of the user change. The existing ECDH key exchange is similar to *KKE*, except that *KKE* provides a key exchange with authentication. In *DMA*, every key derivation user in smart vehicles verifies the identity and other attributes of its key derivation partner and then uses *KKE* to generate authenticated preshared key before deriving a shared secret session key. Note that the ECDH key exchange is used as an underlying cryptographic primitive in F_{CC} to support key exchange. A real-world 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 F, i.e., $P|F_{CC}[c] \leq_R F$. Once $P|F_{CC}[c] \leq_R F$ is proven using the extended composition theorem (cf. Section 3), F_{CC} can be replaced with its realization P_{CC} (see below), where all the ideal operations provided by F_{CC} are replaced by the corresponding real operations. On a high level, the main guarantees provided by F_{CC} are as follows: i) for dynamic mutual multi-factor authentication, F_{CC} guarantees that only honest users can be authenticated and on a per-session basis; ii) for key exchange, F_{CC} guarantees that only honest and authenticated users can exchange preshared keys; and iii) for key derivation, F_{CC} guarantees that only honest and authenticated owners of a preshared key can get access to the shared secret session keys derived from it.

Formally, F_{CC} is a machine with n I/O tapes and a network tape, where the I/O tapes represent different roles in a real-world protocol while the network tape is used for communicating with the adversary. In every run of a system that uses F_{CC} for 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} in smart vehicles is identified by a tuple (ID, sid, l), where ID is the user identity, sid is a session identifier, and l is the role/tape which connects the user to F_{CC}, ID, and sid, which is chosen and managed by real-world 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 say that a message m is a restricting message when the message $(\text{Respond}, restricted, m)$ is sent on the network to ensure that the adversary always responds to all requests and cannot interfere with the run of F_{CC}.

We parameterize F_{CC} with an ECDH domain parameters $EDPGen(1^\eta)$ algorithm and a timestamp function f that takes a message m and outputs a timestamp $f(m) = t$. We require that f is efficiently computable and provides a regular timestamp for timing executions of cryptographic operations as well as sending, receiving and transmitting messages. $EDPGen(1^\eta)$ is used to generate the ECDH domain parameters, where η is a security parameter. The algorithm 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 random values, cryptographic proofs, and cryptographic timestamps in a database that is created by a smart vehicle trusted entity (STE). It also uses the database to prevent any collision (i.e., if a new unknown private key is provided, then it does not already exist in the database) or guessing (i.e., if a new known private key is provided, then it already exists in the database). We assume that F_{CC} synchronizes its clock with that of the STE's database using a clock discipline algorithm [29], which corrects and updates clock time for timestamp accuracy. We note that: (I) As the clock synchronization can be carried out prior to a protocol activation, this makes it difficult for an attacker to bypass the clock synchronization component. (II) The clock discipline algorithm is supported by Network Time Protocol (NTP), which uses an authentication mechanism to prevent manipulations by the attacker.

F_{CC} executes $EDPGen(1^\eta)$ upon its first activation and store the generated domain parameters (p, a, b, G, n, h). Furthermore, we expect F_{CC} to receive users' identities (Users, IDs). When the activation is complete, F_{CC} either returns control to the adversary if the first message/response was received on a network tape or continues to process the original message that activated it if the first message/response was received on the I/O tape. The cryptographic commands that F_{CC} provides to a key derivation initiator (ID, sid, l) and a key derivation responder (ID', sid', l') on their respective I/O interfaces are as follows: (I) **Generate a fresh ephemeral random value** (GetRV), which is provided to both (ID, sid, l) and (ID', sid', l') and returns (RV, r, t) to the user at the end of its execution, where r is an ephemeral random value and t is timestamp of computing r. (II) **Generate a fresh ephemeral private and compute its corresponding ephemeral public key** $(\text{CompEPuK}, r, t)$, which is provided to both (ID, sid, l) and (ID', sid', l') and returns (EPuK, ptr_i, Q) to the user at the end of its execution, where the pointer ptr_i points to an ephemeral private key d and $Q = d.G$ is the corresponding ephemeral public key of d. (III) **Generate a fresh preshared key** $(\text{GenPSK}, ptr_i, r, Q, r', Q')$, which is provided to both (ID, sid, l) and (ID', sid', l') and returns $(\text{PSKPointer}, ptr_{ii})$ to the user at the end of its execution, where the pointer ptr_{ii} points to a new preshared key $k = F_\eta(H((d.Q'), (r', r)))$ and k is computed using SHA-256 hash algorithm $H(.)$ and PRF F, and $d.Q'$ represents an ECDH key. (IV) **Shared secret session key derivation** $(\text{DeriveSKey}, ptr_{ii}, r, r')$, which is provided to only (ID, sid, l) and returns $(\text{SKeyPointer}, ptr_{iv})$ to the user at the end of its execution, where the pointer ptr_{iv} points to a shared secret session key $k_i = F'_\eta(H(k.r.r'))$, and r' is an ephemeral random values of another user. (V) **Compute a cryptographic proof and a cryptographic timestamp** $(\text{CompP\&E}, ptr_{iv}, r, r')$, which is provided to only (ID, sid, l) and returns $(\text{P\&E}, proof, x, ctimestamp, t_3)$ to the user at the end of its execution, where $proof = MAC_{k_i}(x)$ is a cryptographic proof computed at timestamp t_3 and it uses an AES 128-bit encryption algorithm $Enc(.)$ and 256-bit MAC algorithm $MAC(.)$, $ctimestamp = t_3.H(k_i, r, r')$ is a cryptographic timestamp, and $x = Enc_{k_i}(k_i, r, r')$. (VI) **Verify a cryptographic proof and validate a cryptographic timestamp** $(\text{VerP/ValT}, ptr_{iv}, proof, x, ctimestamp, t_3)$, which is provided to only (ID', sid', l') and returns $(\text{Validation}, t_6)$ to the user at the end of its execution, where $proof$ is verified via $VMAC_{k_i}(proof, x) = 1$?

at a timestamp t_6 and $(k_i, r, r^{'}) = Dec_{k_i}(x)$, and $ctimestamp$ is validated via $t_3.H(k_i, r, r^{'}) = ctimestamp$. The description of the six above commands is provided in Appendix A. Note that the combination of authentication mechanisms in GetRV, CompEPuK, GenPSK, DeriveSKey, CompP&E, and VerP/ValT commands represent the *DMA*, while the combination of authentication and key exchange mechanisms in CompEPuk and GenPSK commands represent the *KKE*.

A user of F_{CC} can request for the corruption status of any of its pointers or keys and the environment can request whether the keys of a user is stored as corrupted. In the real and ideal worlds, the environment ensures that a key has the same corruption status. F_{CC} assumes that every user knows the inputs to be provided for every command. One could easily see that if the right data is not provided to execute a command, such command cannot be executed.

We now construct a realization P_{CC} of F_{CC} by using standard cryptographic schemes to implement all commands/operations of F_{CC}. 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 timestamp function f with similar properties as F_{CC}, three encryption schemes $\sum_{authenc}$, $\sum_{unauthenc}$, and $\sum_{pubenc}$ for authenticated symmetric key encryption, unauthenticated symmetric key encryption, and public key encryption, respectively, a mac scheme $\sum_{MAC}$, two families of PRF $F = \{F_{\eta}\}_{\eta \in N}$ and $F^{'} = \{F^{'}_{\eta}\}_{\eta \in N}$ that take key(s) and salt as input and output a key (see [19] for formal definition of these primitives).

To activate P_{CC} for the first time, we expect P_{CC} to receive some message m and then initializes itself by executing $EDPGen(1^{\eta})$ and storing the results before processing m. P_{CC} maintains information in a similar way as F_{CC}. It keeps track of the ephemeral random values, keys, and pointers to (i) determine which type of cryptographic primitive to execute with a given ephemeral random value, key, and pointer, for example, F is used for deriving preshared keys from ECDH keys and salts, while $F^{'}$ is used for deriving shared secret session key from preshared keys and salts and to (ii) provide strong security guarantees that all the values and keys are fresh. The implementation of the F_{CC}'s commands in P_{CC} is provided in Appendix B. Note that the commands GetRV, CompEPuK, GenPSK, DeriveSKey, CompP&E, and VerP/ValT returns (r, t), (ptr_i, Q), ptr_{ii}, ptr_{iv}, $(proof, x, ctimestamp, t_3)$, and t_6, respectively to the user executing the command at the end of its execution.

We now show that P_{CC} realizes F_{CC}. To show this, we want to state and prove our theorem that $P_{CC}[c]$ realizes $F_{CC}[c]$. We want to use standard cryptographic assumptions for this proof as these assumptions provide security guarantees. Additionally, we want to use the Decisional Diffie-Hellman (DDH) assumption [28], [1] to prove that the simulator in this proof $P_{CC}[c] \leq_R F_{CC}[c]$ provides k_i and $k_i^{'}$ for uncorrupted and corrupted shared secret session key, respectively. To capture the expected properties of P_{CC}/F_{CC}, we restrict the environment in this proof not to cause commitment problem [7] and key cycles [4], i.e., it's a well-behaved environment, as many real-world protocols satisfy the conditions of the well-behaved environment and thus if these protocols are analyzed using F_{CC}, then F_{CC} can be replaced with P_{CC} afterwards.

We introduce a machine M^* to ensure that the conditions are satisfied by the environment. In the following theorem, M^* is inserted between the I/O interface of P_{CC}/F_{CC} and the environment. M^* forwards all messages while checking that the conditions are satisfied. If any of the two conditions is not met, M^* stops the forwarding of messages and blocks all communications going forward. We now obtain the following theorem:

Theorem 3. *Let $\sum_{authenc}$, $\sum_{unauthenc}$, $\sum_{pub}$ be encryption schemes, $\sum_{MAC}$ be a MAC scheme, $EDPGen(1^{\eta})$ be an algorithm as above, $F^{'}$ be a family of pseudo-random functions, and F be a family of pseudorandom functions for EDPGen. Let P_{CC} be parameterized with these algorithms. Let F_{CC} be parameterized with an ECDH domain parameters algorithm EDPGen and a timestamp function f. Let c be a sequence of cryptographic operations of P_{CC} and F_{CC} such that every operation c_i in c always rely on the preceding operation c_{i-1}. Then, the following holds true*

$$M^*|P_{CC}[c] \leq_R M^*|F_{CC}[c]$$

if $P_{CC}[c_i(c_{i-1})] \leq_R F_{CC}[c_i(c_{i-1})]$, $\sum_{MAC}$ is IND-CPA and INT-CTXT secure, $\sum_{unauthenc}$ and $\sum_{pub}$ is IND-CCA2 secure, $\sum_{MAC}$ is UF-CMA secure, EDPGen always outputs random primes for field order p and groups with $n \geq 2$ and such that the DDH assumption holds true for EDPGen.

We provide a proof sketch of this theorem in Appendix C.

4.2 Ideal Functionality for Mutual Authentication and Key Derivation

The main idea of our ideal functionality for mutual authentication and key derivation F_{MKD} is that authenticated users can send key derivation requests to F_{MKD} to start a mutual authentication and key derivation by deriving a session key, a cryptographic proof, and a cryptographic timestamp to ensure relay resilience. Formally, F_{MKD} is a machine that has two I/O tapes, one network tape, and another two I/O tapes for connecting to F_{CC}, which is used by F_{MKD} for as a subroutine for cryptographic operations. We parameterize F_{MKD} with symmetric key of type $s_{key} \in \{authenc - key, unauthenc - key\}$, maximum end-to-end delay (EED) for messages sent from a key derivation initiator to a responder $max.eed1$, maximum end-to-end delay for messages sent from a key derivation responder to the initiator $max.eed2$, and expected time of completing a mutual authentication and key derivation session $et.mkd$, which refers to the amount of time expected for all cryptographic operations associated with computing and verifying a cryptographic proof to be executed. s_{key} represents the type of key that can be output after successful mutual authentication and key derivation. Note that maximum EED refers to the average time taken by the messages to reach the destination from the source. Similar to F_{CC}, the sessions for all users are handled by F_{MKD} and every user participating in the mutual authentication and key derivation can be identified by (ID, sid, l), where l specifies the role of the user, i.e., $l \in \{A, B\}$. F_{MKD} also uses the ITE's database for any check or verification. Furthermore, messages from/to every I/O tape are prefixed with (ID, sid) to prevent any user from claiming to be another user.

F_{MKD} manages sessions of users and global sessions: A user (ID, sid, l) can send start a mutual authentication and key derivation. The adversary/simulator groups sessions of an initiator A and a responder B to form global sessions. F_{MKD} makes two assumptions when analyzing timestamps of messages: i) Her clock and F_{CC}'s clock, as well as the initiator clock and responder clock, move-in relative synchrony while users are in a global session; and ii) network speed is symmetric, i.e., the data speed from initiator to the responder as well as the data speed from the responder back to the initiator are the same. F_{MKD} maintains several key derivation states *restricted*, $startedMKD$, $finishedMKD$, $closedMKD$, and *corrupted* for mutual authentication and key derivation and provides seamless transitions between states to prevent broken authentication. The initial state of every user (ID, sid, l) is set as *restricted*. In F_{MKD}, we keep the information of every user in an active session as a secret to support the security of mutual authentication and key derivation. Note that one can observe that a session is active by monitoring the network of the user. Furthermore, as long as a user is not in the state *restricted* or *corrupted*, F_{MKD} answers such a user's request without contacting the adversary. However, if the user is in the state *restricted* or *corrupted*, F_{MKD} asks the adversary about any request received by sending a restricting message.

To provide relay resilience, F_{MKD} uses the EED of a message or a data packet from an initiator to a responder $eed1$, EED of a data packet from the responder to the initiator $eed2$, the computation time of cryptographic operations at the responder $cce2$, and computation time of cryptographic operations at the initiator $cce1$. Note that the computation times of the above operations are not known in advance and as such an attacker cannot carry out attacks that change these times and thus cannot mount Denial of Service (DoS) attacks for the users. In a scenario where the attacker has estimated these times before doing the DoS attacks, we use $Enc(.)$ and $MAC(.)$ to authenticate timestamps (see below) and provide timestamps confidentiality, integrity, and authenticity thereby preventing the attacker from carrying out such attacks. The conditions for relay resilience in the smart vehicles are as follows: i) $eed1$ is directly proportional to $eed2$ based on $max.eed1$ and $max.eed2$, respectively; ii) $eed2$ is directly proportional to $eed1$ based on $max.eed2$ and $max.eed1$, respectively; iii) $cce2 \approx cce1$; and iv) $cce1 + cce2 + eed1 + eed2 \leq et.mkd$. If the above conditions hold, this shows that the mutual authentication and key derivation is complete and thus provides relay resilience (cf. Section 3.3). Note that if the attacker introduced a DoS attack by delays, the above conditions cannot be satisfied and thus such an attack is mitigated. The operations provided by F_{MKD} to mutual authentication and key derivation protocols are as follows:

- A user (ID, sid, l) with state *restricted* can start a mutual authentication and key derivation by sending a message $m = (\text{InitKE}, ID^{'}, m_i)$, where $ID^{'}$ is user identity of the intended key derivation responder and m_i is a random bit string that can be used by the realization in the key derivation protocol. Upon receiving m, F_{MKD} checks that both users are not yet part of a global session and ID and $ID^{'}$ exist in the database and sets $responder(ID, sid, l) = ID^{'}$ if the checks succeed. Then, F_{MKD} forwards $(m, (ID, sid, l))$ to the adversary via a restricting message. The adversary responds by sending $(\text{CreateGroupSession}, (ID_A, sid_A), (ID_B, sid_B))$ to F_{MKD}. Then, F_{MKD} sets the state of the users to $startedMKD$, stores that (ID_A, sid_A, A) and (ID_B, sid_B, B) are in the same global session, and sends *Okay* to the adversary.
- A user (ID, sid, l) with $state(ID, sid, l) = startedMKD$ can use $(\text{CompEED1}, t_1, t_2)$ to request F_{MKD} to compute the EED $eed1$ of a message using receiving time t_2 and sending time of the message t_1. In this case, F_{MKD} first computes $eed1 = t_2 - t_1$ and checks whether $eed1 \leq max.eed1$. If the check succeeds, F_{MKD} returns $(\text{EED1}, eed1)$ to the user. Otherwise, it returns $(\text{EED1}, restricted)$ to the user.
- A user (ID, sid, l) with $state(ID_A, sid_A, A) = startedMKD$ can get secured timestamps by sending a request $(\text{GetTimestamps}, y, z, ptr_{ii})$ to F_{MKD}. If k is recorded in the database as a preshared key of user ID_A via the pointer ptr_{ii}, F_{MKD} uses the $Dec(.)$ and $VMAC(.)$ algorithms provided by the adversary to decrypt $y = Enc_k(ctimestamp, t_2, t_3)$ and verify $z = MAC_k(y)$, respectively, and returns $(\text{Timestamps}, ctimestamp, t_2, t_3)$ to the user. Furthermore, the user can also use F_{MKD} to get access to F_{CC} commands such as GetRV, CompEPuK, GenPSK, DeriveSKey, and VerP/ValT. As soon as a response of VerP/ValT execution is received, F_{MKD} sets $state(ID_A, sid_A, A) := finishedMKD$.
- A user (ID, sid, l) with $state(ID_B, sid_B, B) = startedMKD$ can use F_{MKD} to access F_{CC} commands such as GetRV, GetEPuK, GenPSK, DeriveSKey, and CompP&E. F_{MKD} forwards the responses it received from F_{CC} to the user while internally keeps track of all values, keys, and pointers associated with the user. Furthermore, the user can use $(\text{SecureTimestamps}, t_2, t_3, ctimestamp, ptr_{ii})$ to request F_{MKD} to secure the timestamps related to computation time of cryptographic operations using timestamps t_2 and t_3, cryptographic timestamp $ctimestamp$, and the preshared key k to which the pointer ptr_{ii} points. If k is recorded in the database as a preshared key of user ID_B, F_{MKD} uses the $Enc(.)$ and $MAC(.)$ algorithms provided by the adversary to compute $y = Enc_k(ctimestamp, t_2, t_3)$ and $z = MAC_k(y)$, respectively. Then, F_{MKD} returns $(\text{Timestamps}, y, z)$ to the user and sets $state(ID_B, sid_B, B) := finishedMKD$.
- A user (ID, sid, l) with $state(ID_A, sid_A, A) = finishedMKD$ can use $(\text{CompEED2}, t_4, t_5)$ to request F_{MKD} to compute the EED $eed2$ of a data packet from its key derivation partner (i.e., responder) using receiving time t_5 and sending time of the message t_4. In this case, F_{MKD} computes $eed2 = t_5 - t_4$, and then checks whether $eed2 \leq max.eed2$. If the check succeeds, F_{MKD} returns $(\text{EED2}, eed2)$ to the user. Otherwise, it returns $(\text{EED2}, restricted)$ to the user.
- A user (ID, sid, l) with $state(ID_A, sid_A, A) = finishedMKD$ can use $(\text{CompCCE}, t_0, t_1, t_2, t_4, t_5, t_6)$ to request F_{MKD} to verify the computation time of cryptographic operations at the responder $cce2$ and computation time of cryptographic operations at the initiator $cce1$ using timestamps t_0, t_1, t_2, t_4, t_5, and t_6, where $t_0 = t_A$ is generated from execution of GetRV command. In this case, F_{MKD} computes $cce2 = t_4 - t_2$ and $cce1 = (t_6 - t_5) + (t_1 - t_0)$, and verifies that $cce2 \approx cce1$.

If the verification succeeds, F_{MKD} returns (CCE, $cce2, cce1$) to the user. Otherwise, it returns (CCE, $restricted$) to the user.

- A user (ID, sid, l) with $state(ID_A, sid_A, A) = finishedMKD$ can use (VerETimeMKD, $cce2, cce1, eed1, eed2$) to request F_{MKD} to verify the expected time of completing a mutual authentication and key derivation session $et.mkd$ using $cce2$, $cce1$, $eed1$, and $eed2$. Upon receiving this request, F_{MKD} computes $mkd = cce2 + cce1 + eed1 + eed2$ and verifies $mkd \leq et.mkd$. If this verification succeeds, F_{MKD} returns $Okay$ to the user and sets $state(ID_A, sid_A, A) := closedMKD$ to provide relay protection. Then, it notifies the adversary via a restricting message (ClosedSession, (ID, sid, l)) and forwards the response to the user. Thus, the user loses access to all values, keys, pointers, and commands.
- A user (ID, sid, l) with $state(ID_B, sid_B, B) = finishedMKD$ can use F_{MKD} to close her session by sending a request (CloseSession, (ID, sid, l)). Upon receiving this request, F_{MKD} sets $state(ID, sid, l) := closedMKD$, notifies the adversary via a restricting message (CloseSession, (ID, sid, l)) and sends $Okay$ to the user after receiving the adversary's response. As the session has been closed, the user no longer has access to keys, pointers, values, and commands.
- A user (ID, sid, l) with state $closedMKD$ or $restricted$ can be corrupted by the adversary since the user has no access to any values, keys, pointers, and commands. Upon receiving m = (Corrupt, (ID, sid, l)), F_{MKD} sets $state(ID, sid, l) := corrupted$. Furthermore, the user can ask for its corruption status and then F_{MKD} sends a restricting message (CorruptionStatus, (ID, sid, l)) to the adversary. In this case, F_{MKD} forwards the response to the user. Hence, these two scenarios represent a simple corruption model of F_{MKD}.

F_{MKD} models relay resilience upon the successful execution of VerETimeMKD since the attributes of the users and knowledge of computation time of cryptographic operations and cryptographic timestamps hold. Note that since the adversary does not have access to any keys after mutual authentication and key derivation, F_{MKD} models perfect forward secrecy to enhance relay resilience.

An uncorrupted user can be paired with a corrupted user by the adversary, however, the corrupted user will not get access to the session key in F_{CC}. Furthermore, we restrict the corruption model to improve the use of F_{MKD} by real-world protocols in smart vehicles. This restriction does not affect the expressivity of our functionality as session keys derived from many real-world protocols are usually short-lived and thus the chance for corruption is negligible.

5 MUTUAL AUTHENTICATION AND KEY DERIVATION PROTOCOL (MKD)

In this section, we present our mutual authentication and key derivation protocol MKD with relay-resilience security properties such as mutual multi-factor authentication, key exchange, and key derivation as depicted in Figure 3. MKD is the first protocol to use a full-fledged formal framework, i.e., Crypto-Chain, for relay resilience in smart vehicles. Formally, MKD uses F_{CC} to perform cryptographic operations and realizes F_{MKD}. We use two machines $M_{A(MKD)}$ and

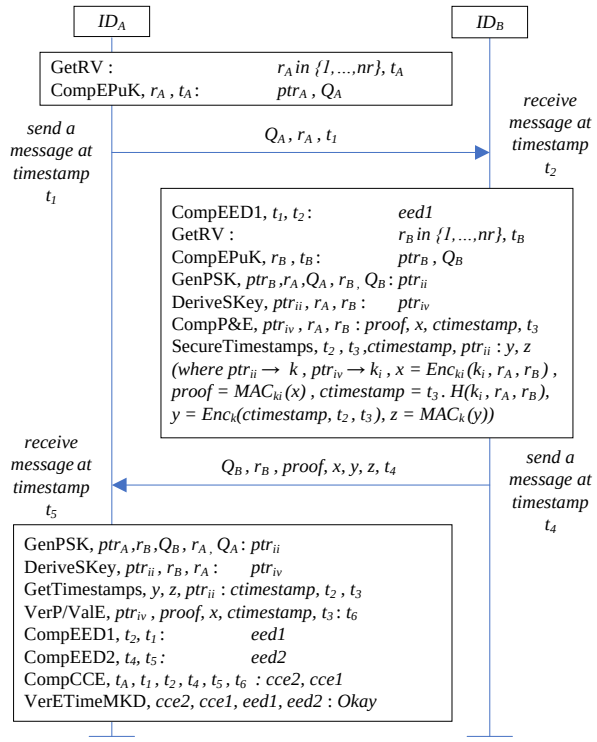

Figure 3: Mutual Authentication and Key Derivation Protocol (MKD).

$M_{B(MKD)}$ to model the role of key derivation initiator A and responder B of our protocol. Only one instance of $M_{A(MKD)}/M_{B(MKD)}$ for every user (ID, sid) is available in the run of MKD to execute the protocol from Figure 3. At the end of the protocol run, the instances generate a preshared key k, derive a shared secret session key k_i, deliver pointers to these keys, and compute cryptographic proof $proof$ and cryptographic timestamp $ctimestamp$. Furthermore, the instances allow the user to use F_{CC} to perform cryptographic operations with the keys in an ideal manner. We provide the theorem and proof of MKD in Appendix D, which shows that MKD is a secure universally composable relay resilience mutual authentication and key derivation protocol.

6 IMPLEMENTATION AND ANALYSES OF MKD

In this section, we conduct some experiments to implement MKD on Tmote Sky mote sensors [10], which are widely used in smart vehicles, and then analyse the efficiency and relay resilience capabilities of MKD. Tmote Sky mote sensors such as CM5000, XM1000, and CM4000 are low power wireless sensors that are interoperable with many IEEE 802.15.4 devices, offer a high data rate, and support stability and implementation of time synchronization. We implement MKD on the sensors to provide insights about its actual deployment in the smart vehicles as well as demonstrate the efficiency of Crypto-Chain. We connect the sensors to a MacBook

Table 1: Computation time of cryptographic algorithms in Crypto-Chain

Cryptographic Algorithm	Tmote Sky mote sensors
AES 128-bit symmetric encryption	0.00170 s
AES 128-bit symmetric decryption	0.00167 s
SHA-256 hash algorithm	0.0091 s
160-bit ECC point multiplication	1.0400 s
keyed-Hash MAC-SHA256	0.0183 s
32-bit random number	0.0073 ms

Pro (2.3Ghz, Intel Core i5 processor, 8GM memory) for our implementation as follows. Firstly, we implement Crypto-Chain's cryptographic algorithms such as the AES 128-bit encryption algorithm. Note that the cryptographic algorithms are lightweight and are used in this paper to support the practical implementation of Crypto-Chain in smart vehicles. The components used in our experiment are provided in Appendix E. All programs implementing the cryptographic algorithms are written in nesC language [15] and we implement the algorithms under the TinyOS operating system [16]. The computation time of each cryptographic algorithm is presented in Table 1 for brevity. The breakdown of the number of bits and computation time of cryptographic operations in our experiments are given as follow: (I) Generate an ephemeral random value and public key – 576 bits for 1.04002 s; (II) Compute End-to-End Delay - 64 bits for 0.0146 ms; (III) Generate preshared secret key - 448 bits for 0.01823 s; (IV) Compute a cryptographic timestamp - 960 bits for 0.03824 s; (V) Validate a cryptographic timestamp - 960 bits for 0.03819 s; (VI) Secure timestamps - 768 bits for 0.02909 s; (VI) Get timestamps - 768 bits for 0.02908 s; and (VII) Verify expected time of completing MKD - 224 bits for 0.05122 ms. Lastly, we assess the maximum computational cost required for MKD execution between two users (i.e., initiator and responder) and our results show the following: (I) The responder requires $\approx$ 1.1255 s, which can also be derived using $(t_4 - t_2)$ as per the above computation time; and (II) The initiator requires $\approx$ 1.1256 s, which can also be derived using $((t_6 - t_5) + (t_1 - t_0))$ as per the above computation time. Thus, the computation time of cryptographic operations at the responder $cce2$ is $\approx$ 1.1255 s and the computation time of all cryptographic operations at the initiator $cce1$ is $\approx$ 1.1256 s.

We simulate MKD using the widely accepted network simulation tool, Network Simulator 3 (NS-3) [9] to measure its EED and further validate its relay resilience. The details of the simulation parameters we used in NS-3 are as follows: i) Platform is Ubuntu 18.04 LTS; ii) Communication medium is Wi-Fi; iii) Channel model is P2P; iv) Transport layer is UDP; v) constant speed is $\approx 3 * 10^8\ ms^{-1}$ (speed of light); and vi) distance is 100 m. We also consider other standard NS-3 parameters such as measuring network protocols performance using a flow monitor. The maximum size of message to be transmitted by the key derivation initiator and responder is $(Q_A, r_A, t_1) \approx 384\ bits$ and $(Q_B, r_B, proof, x, y, z, t_4) \approx 1,760\ bits$, respectively. The EED of MKD is $\approx$ 8.2573 ms. We can see that the EED is less than our maximum 10 ms latency target of smart vehicles applications systems such as cooperative driving and automated overtaking [24, 33]. Hence, MKD is suitable and fit for the smart vehicles and it further validates the efficiency of Crypto-Chain (based on the transmitted messages). Furthermore, we consider the impact of different distances on EED and our results show that longer distances increase the EED. We recommended that each of the initiator and responder should have a maximum EED (supported by the size of the message) to verify the EED of the message received and the responder should verify the computation time of cryptographic operations (see below) to prevent an attacker from tampering or shortening the distance of communication.

We utilize the EED result and computation time of cryptographic operations to further mitigate relay attacks using MKD. Taking values from our simulation, $eed1$, which represents the EED of the data packet of size 384 $bits$ from ID_A to ID_B, is 7.8374 ms, and $eed2$, which represents the EED of the data packet of size 1,536 $bits$ from ID_B to ID_A is 7.9066 ms. Recall, $et.mkd = cce1 + cce2 + eed1 + eed2$ (cf. Section 4). Hence, $et.mkd$ = 1.1255 s + 1.1256 s + 7.8374 ms + 7.9066 ms = 2.2668 s. Note that the EED increases as the size of a message increases. We consider the above results as the actual EEDs required for mutual authentication and key derivation based on our simulation settings, i.e., $max.eed1$ = 7.8374 ms, $max.eed2$ = 7.9066 ms, and $et.mkd$ = 2.2668 s.

To show relay resilience using MKD, we use one our sensors (i.e., CM4000) as an attacker I to relay data packets between ID_A (i.e., XM1000) and ID_B (i.e., CM5000) in our simulation. The $eed1_I$ with route $ID_A \rightarrow I \rightarrow ID_B$ is 8.0086 ms and $eed2_I$ with route $ID_B \rightarrow I \rightarrow ID_A$ is 8.1293 ms. This shows that: (A) $eed1_I > max.eed1$; (B) $eed2_I > max.eed2$; (C) $eed1_I$ is not directly proportional to $eed2_I$ based on $max.eed1$ and $max.eed2$, respectively; (D) $eed2_I$ is not directly proportional to $eed1_I$ based on $max.eed2$ and $max.eed1$; and (E) $cce1 + cce2 + eed1_I + eed2_I > et.mkd$. Thus, we conclude that (i) since $eed1_I > max.eed1$, CompEED1's execution at ID_B will return restricted to provide relay resilience and that (ii) suppose $eed1_I < max.eed1$ and any of "B", "C", and "D" above holds true, the mutual authentication and key derivation session will also be restricted at ID_A to provide relay resilience. The above simulation and conclusion further validates the relay resilience capabilities of Crypto-Chain based on the transmitted messages and EED values. Note that: (I) Our implementation and analyses focus on demonstrating the efficiency of our framework via satisfying our 10 ms latency target and illustrating its relay resilience capability. (II) Our framework can be supported by an NTP software that operates in each hardware. Any software/hardware modifications will be based on the NTP to provide the highest clock accuracy.

7 CASE STUDIES

In this section, we carry out two case studies to show the application of our framework.

7.1 Megamos Crypto

The Megamos Crypto [5] is a symmetric cryptosystem. It is one of the most widely deployed smart vehicle immobilizer systems used in many Honda, Audi, Fiat, and Volkswagen cars. This system is designed to act as a vehicle anti-theft solution between a vehicle and a car key. These devices have a microprocessor chip that incorporates the Megamos Crypto. We extracted and described the authentication protocol in the Megamos Crypto in Figure 1. It is

based on a pre-established shared secret key and PRF. We model the Megamos Crypto in our framework using two machines $M_{A(MC)}$ and $M_{B(MC)}$ to model the initiator (i.e., the vehicle) and responder (i.e., the car key) roles, respectively. These machines have a similar I/O interface as F_{MKD} and each of them has a network tape. They use F_{CC} as a function to perform cryptographic operations. In every run of the Megamos Crypto, there is only one instance of $M_{A(MC)}/M_{B(MC)}$ per user (ID, sid) executing the protocol as depicted in Figure 1 to negotiate a random number for authenticating the identity of the responder.

The Megamos Crypto does not mitigate relay attacks and thus cannot realize F_{MKD}. To see this, we consider the following setting: an honest initiator authenticates an honest responder that provided a random value r_B. The responder instance might have received a genuine public key and a random value, say (Q_A, r_A) in the first message of the protocol. The Megamos Crypto does not guarantee that (i) the initiator and responder sent (Q_A, r_A) and r_B, respectively, and that (ii) an attacker did not relay (Q_A, r_A) and r_B. The attacker can relay r_B and (Q_A, r_A) as a relay attack does not manipulate the messages transmitted between the initiator and responder. Thus, we have no security guarantee for the transmitted messages and the attacker can easily let the initiator and responder accept r_B and (Q_A, r_A), respectively. While this is not a direct attack on the protocol, it shows that the security of the Megamos Crypto is not sufficient to mitigate relay attacks. The fixes for this problem in our setting are given as follows: i) enhance the first message of the protocol with a timestamp, i.e., introduce a timestamp t_1 and equip the protocol with CompEED1 operation of F_{MKD} to compute EED; and ii) equip the protocol with all the commands of F_{CC} and VerETimeMKD operation of F_{MKD} to avoid the reliance on pre-established shared secret key for authentication in the Megamos Crypto and provide mutual authentication and key derivation security guarantees. Thus, using F_{MKD} provides an *Enhanced Megamos Crypto* as described in the above fixes. The *Enhanced Megamos Crypto* theorem is provided in Appendix F.

7.2 Hitag-AES/Pro

The Hitag-AES/Pro [31] is a smart vehicle immobilizer transponder based on AES 128-bit encryption algorithm. We extract the authentication protocol in Hitag-AES/Pro as depicted in Figure 4. It is meant to provide mutual authentication during communication between a reader and a transponder using a pre-established shared secret key. We model the initiator (i.e., the reader) and responder (i.e., the transponder in a car key) in a similar way to the Megamos Crypto, except that we have two machines $M_{A(HP)}$ and $M_{B(HB)}$ modelling the initiator and responder, respectively, of the Hitag-AES/Pro. At the end of the execution of this protocol, the instances exchange messages to authenticate each other in an ideal manner. While Hitag-AES/Pro is widely used in the automotive industry, implementing AES 128-bit encryption algorithm does not guarantee relay resilience as a relay attack can still be performed irrespective of the cryptographic algorithms deployed. Thus, Hitag-AES/Pro does not realize F_{MKD}. To show this: we consider a setting where the honest initiator decrypts the fourth message m of the protocol, i.e., $Dec_k(m) = Dec_k(Enc_k(n_R, PWD_B))$, received from the honest responder as per Figure 4. If m was sent or not sent by the responder,

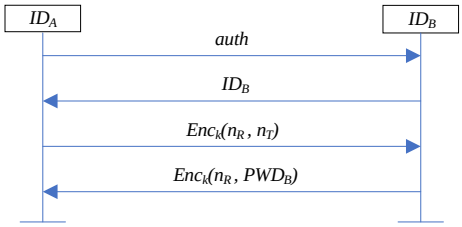

Figure 4: Hitag-AES/Pro Authentication Protocol. Abbreviations: $auth$ – authentication request, n_R, n_T – random nonces, k – pre-established shared secret key, PWD_B – responder password.

the initiator will still decrypt the message and act according to the protocol execution because the $Enc_k(.)/Dec_k(.)$ do not guarantee any relay resilience and we have no relay resilience guarantee for $Enc_k(.)$, $Dec_k(.)$, and m. Similar to the Megamos Crypto, this is not a direct attack on the protocol but rather shows that the security of $Enc(.)/Dec(.)$ is not sufficient to prove that a protocol is resilience against a relay attack.

A fix for this problem in our setting is to have a cryptographic proof during $Enc_k(.)$ and verify the proof during $Dec_k(.)$ using ComP&E and VerP/ValT commands of F_{CC}, respectively, and then use VerETimeMKD operation of F_{MKD} to validate the expected time of completing the protocol execution. This allows the enhancement of the protocol using F_{MKD} which yields an *Enhanced Hitag-AES/Pro*. Other commands of F_{MKD} and F_{CC} can also be used in the protocol. This *Enhanced Hitag-AES/Pro* theorem is provided in Appendix G.

8 RELATED WORK

The role of relay attacks in exploits against cryptographic protocols in smart vehicles has been widely recognized with the practical implementation of the attacks in [14, 34, 36]. We adopt the practical implementation in [36] and then focus on proving relay security as a form of relay resilience solution and enhancement to the existing related work. Verdult et al. [35] discussed effective authentication protocol for mitigating relay attacks. The protocol relies on the AES 128-bit encryption algorithm, which is already implemented in ATA5795C and Hitag-AES/Pro. However, the AES 128-bit encryption algorithm does not solve the problem of relay attacks as encrypted data can be relayed without decrypting it, i.e., relay attacks are independent of the cipher used in designing the protocol. Similarly, using authentication for relay resilience is frequently discussed in [25, 26, 39] and poor implementation of authentication protocols can expose the applications to relay attack due to implementation errors such as lack of key update and randomization during authentication.

Verdult et al. [36] proposed some measures to mitigate relay attacks. These measures include randomly generating secret keys, redesigning weak ciphers, and using immobilizers that implement AES 128-bit encryption algorithm. Relay attacks can still be performed in the presence of secret keys and AES 128-bit encryption

algorithm because they do not depend on manipulation or interpretation of plaintexts or ciphertexts. One of the key countermeasures presented in [36] to mitigate relay attacks is implementing asymmetric cryptography for data authentication. However, authentication alone is not sufficient to mitigate a relay attack. Wan et al. [38] proposed to use key derivation schemes for car appliances to provide relay resilience and prevent the use of vehicles without permission. Key derivation countermeasures against relay attacks can inadvertently increase attack vectors. For instance, derived keys that rely on pre-established secret keys and weak cryptographic primitives introduce weak keys flaw to the car appliances.

Francillon et al. number[14] proposed mitigation measures to prevent relay attacks on passive keyless entry and start (PKES) systems in smart cars. These measures include shielding the key in the key case, removing the battery from the key, access control restrictions, temporary disabling of the PKES systems via software modification, and removing the battery from the key fob via hardware modification. Overall, their solution relies on RF distance bounding, which has been extensively studied in [20, 21] to provide accurate measurement of distance for relay attack prevention. However, the solution affects the availability of the PKES systems as the operation of the systems is disabled to execute countermeasures, and the solution lacks cryptographic soundness to implement all the basic relay resilience security properties such as authentication. Note that a number of distance bounding protocols have been proposed to mitigate relay attacks [17, 30]. However, these protocols are vulnerable to distance hijacking attack [11]. To protect against distance hijacking, the protocols are encouraged to use different incompatible hardware for distance measurement. While this protection measure is applicable in the real world, the dispersed and fast-changing nature of smart vehicles' environments with many different and similar hardware that can communicate with each other makes it difficult for its extensive real-world deployment.

Li et al. [27] proposed a smartphone-enabled user context detection system for relay attack mitigation. The system relies on the direction assumption that a vehicle owner is either walking towards the vehicle or is inside the vehicle to mitigate relay attacks. Considering that relay attacks can be carried out irrespective of the walking direction or position of the vehicle owner, this shows that the system is unable to mitigate relay attacks when the vehicle receives a start instruction from other directions. In this case, the authentication mechanism of the system can be evaded by an attacker. Ahmad et al. [2] presented a secure passive keyless entry and start method using machine learning. While the method supports the detection of relay attacks in a challenge-response setting between a key fob and a vehicle, it does not support mutual authentication between the devices, i.e., the key fob and the vehicle. We note that it is very important to apply mutual authentication between the devices to mitigate authentication evasion. Wang et al. [37] proposed a context-based secure keyless entry system that supports communication over Bluetooth between the key fob and vehicle. The system grants access to the vehicle via a connection established between the key fob and the vehicle using information extracted from the surrounding environment. At the end of the connection establishment, the vehicle generates a signature for the key fob. However, the system relies on public and private keys issued by either an authority or another device for generating the signature thereby presenting a single point of failure and introducing extra communication and computational overheads. Thus, the system does not support key derivation for signature between the key fob and vehicle. Choi et al. [8] proposed a sound-based proximity-detection method for relay attack mitigation on PKES systems. However, the method presents a single point of failure to the systems because of the reliance on a preshared key to perform authentication between the key fob and vehicle. Besides, such a reliance makes the implementation of the method very difficult and expensive in real-world environments.

In this paper, we argue that mutual authentication and key derivation are required to verify user identity and derive a shared secret session key, respectively. Besides, mutual authentication is applied to prevent an attacker from interfering with the transmitted messages between two users. Our framework, which is built irrespective of hardware compatibility, uses the combination of cryptographic operations, knowledge of the computation time of the operations, and cryptographic timestamps to mitigate relay attacks based on universal composability. While many of the above existing solutions did not prove relay security of their work, mitigation solutions, or countermeasures to guarantee that negotiable values, keys, or proofs are resilience against relay attack [35, 36], we prove relay protection for smart vehicles. Our result allows proving relay security as long as we restrict the smart vehicles to our cryptographic operations, knowledge of the computation time of the operations, and cryptographic timestamps. Our framework provides a strong form of relay security for users of smart vehicles. Two users that authenticate each other and derive a shared secret session key in our framework can obtain strong relay resilience guarantees.

9 CONCLUSION AND FUTURE WORK

In this paper, we put forward a universally composable framework, Crypto-Chain, to analyze relay security of many real-world protocols in smart vehicles and provide relay resilience solution with the support of an extended Kusters's universal composition theorem on a fixed number of protocol systems. Crypto-Chain consists of an ideal crypto-chain functionality F_{CC} for cryptographic operations and an ideal functionality for mutual authentication and key derivation F_{MKD}. F_{CC} models various cryptographic primitives, including our new *DMA* and *KKE*. These primitives can be combined and used to negotiate a shared secret session key for relay resilience support. F_{MKD} is supported by F_{CC}'s cryptographic operations, knowledge of the computation time of the cryptographic operations, and cryptographic timestamps.

We have provided a mutual authentication and key derivation protocol *MKD* to further validate the beneficial use of Crypto-Chain for the design of relay resilience cryptographic protocols in the smart vehicles. We have implemented *MKD* and validated its performance and relay resilience capabilities. We further demonstrated the usefulness of Crypto-Chain in two case studies, namely Megamos Crypto and Hitag-AES/Pro. We have uncovered some weaknesses in the relay security of the protocols and used Crypto-Chain to provide relay resilience by enhancing the security of the protocols. In future work, we will apply Crypto-Chain to other real-world protocols and extend it to further mitigate ransomware attacks in smart vehicles.

REFERENCES

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. 2001. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Cryptographers' Track at the RSA Conference*. Springer, 143–158.

[2] Usman Ahmad, Hong Song, Awais Bilal, Mamoun Alazab, and Alireza Jolfaei. 2018. Secure passive keyless entry and start system using machine learning. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 304–313.

[3] Atmel. 2014. ATA5795C - Embedded AVR Microcontroller Including RF Transmitter and Immobilizer LF Functionality for Remote Keyless Entry. http://ww1.microchip.com/downloads/en/devicedoc/atmel-9182-car-access-ata5795c_datasheet.pdf

[4] Michael Backes, Birgit Pfitzmann, and Andre Scedrov. 2008. Key-dependent message security under active attacks - BRSIM/UC-soundness of Dolev-Yao-style encryption with key cycles. *J. Comput. Secur.* 16, 5 (2008), 497–530.

[5] Bicotech. 2019. Megamos Crypto: Read-Write High Security Device - Memory organisation. https://www.bicotech.com/?page=prod_rwprog&lg=en

[6] Jan Camenisch, Robert R Enderlein, Stephan Krenn, Ralf Küsters, and Daniel Rausch. 2016. Universal composition with responsive environments. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 807–840.

[7] Ran Canetti and Marc Fischlin. 2001. Universally composable commitments. In *Annual International Cryptology Conference*. Springer, 19–40.

[8] Wonsuk Choi, Minhye Seo, and Dong Hoon Lee. 2018. Sound-proximity: 2-factor authentication against relay attack on passive keyless entry and start system. *Journal of Advanced Transportation* 2018 (2018).

[9] NS-3 Consortium. 2015. The Network Simulator 3. https://www.nsnam.org/

[10] Moteiv Coorperation. 2005. Tmote Sky: Ultra Low Power IEEE 802.15. 4 Compliant Wireless Sensor Module. *Datasheet: http://www. sentilla. com/pdf/eol/tmote-skydatasheet. pdf* (2005).

[11] Cas Cremers, Kasper B Rasmussen, Benedikt Schmidt, and Srdjan Capkun. 2012. Distance hijacking attacks on distance bounding protocols. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 113–127.

[12] Mahdi Dibaei, Xi Zheng, Kun Jiang, Sasa Maric, Robert Abbas, Shigang Liu, Yuexin Zhang, Yao Deng, Sheng Wen, Jun Zhang, et al. 2019. An overview of attacks and defences on intelligent connected vehicles. *arXiv preprint arXiv:1907.07455* (2019).

[13] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208. https://doi.org/10.1109/TIT.1983.1056650

[14] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2011. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

[15] David Gay, Philip Levis, Robert Von Behren, Matt Welsh, Eric Brewer, and David Culler. 2014. The nesC language: A holistic approach to networked embedded systems. *ACM Sigplan Notices* 49, 4 (2014), 41–51.

[16] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. 2000. System architecture directions for networked sensors. *ACM SIGOPS operating systems review* 34, 5 (2000), 93–104.

[17] Chong Hee Kim and Gildas Avoine. 2009. RFID distance bounding protocol with mixed challenges to prevent relay attacks. In *International Conference on Cryptology And Network Security*. Springer, 119–133.

[18] Ralf Küsters and Max Tuengerthal. 2013. The IITM Model: a Simple and Expressive Model for Universal Composability. *IACR Cryptology EPrint Archive* 2013 (2013), 25.

[19] Ralf Kuesters and Daniel Rausch. 2017. *A Framework for Universally Composable Diffie-Hellman Key Exchange*. Report. Cryptology ePrint Archive, Report 2017/256. https://eprint.iacr.org/2017/256

[20] Marc Kuhn, Heinrich Luecken, and Nils Ole Tippenhauer. 2010. UWB impulse radio based distance bounding. In *2010 7th Workshop on Positioning, Navigation and Communication*. IEEE, 28–37.

[21] P. Syam Kumar, R. Subramanian, and D. Thamizh Selvam. [n.d.]. An Efficient Distributed Verification Protocol for Data Storage Security in Cloud Computing. IEEE, 214–219. https://doi.org/10.1109/ADCONS.2013.46

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

[23] Ralf Küsters and Max Tuengerthal. 2011. Composition theorems without pre-established session identifiers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 41–50.

[24] Maria A. Lema, Andres Laya, Toktam Mahmoodi, Maria Cuevas, Joachim Sachs, Jan Markendahl, and Mischa Dohler. 2017. Business Case and Technology Analysis for 5G Low Latency Applications. *IEEE Access* 5 (2017), 1–1. https://doi.org/10.1109/ACCESS.2017.2685687

[25] Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stüble. 2005. An open approach for designing secure electronic immobilizers. In *International Conference on Information Security Practice and Experience*. Springer, 230–242.

[26] Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stüble. 2006. *Anti-theft protection: Electronic immobilizers*. Springer, 51–67.

[27] Jing Li, Yabo Dong, Shengkai Fang, Haowen Zhang, and Duanqing Xu. 2020. User Context Detection for Relay Attack Resistance in Passive Keyless Entry and Start System. *Sensors* 20, 16 (2020), 4446.

[28] Ueli M Maurer and Stefan Wolf. 1996. Diffie-hellman oracles. In *Annual International Cryptology Conference*. Springer, 268–282.

[29] D. L. Mills. 1998. Adaptive hybrid clock discipline algorithm for the network time protocol. *IEEE/ACM Transactions on Networking* 6, 5 (1998), 505–514. https://doi.org/10.1109/90.731182

[30] Pedro Peris-Lopez, Julio C Hernandez-Castro, Juan ME Tapiador, Esther Palomar, and Jan CA van der Lubbe. 2010. Cryptographic puzzles and distance-bounding protocols: Practical tools for RFID security. In *2010 IEEE International Conference on RFID (IEEE RFID 2010)*. IEEE, 45–52.

[31] NXP Semiconductors. 2011. Hitag-AES/Pro Product Data Sheet. https://www.nxp.com/docs/en/user-guide/UM10277.pdf

[32] Yoshiyasu Takefuji. 2018. Connected vehicle security vulnerabilities [commentary]. *IEEE Technology and Society Magazine* 37, 1 (2018), 15–18.

[33] Andrea Tassi, Malcolm Egan, Robert J. Piechocki, and Andrew Nix. 2017. Modeling and Design of Millimeter-Wave Networks for Highway Vehicular Communication. *IEEE Transactions on Vehicular Technology* 66, 12 (2017), 10676–10691. https://doi.org/10.1109/TVT.2017.2734684

[34] Roel Verdult and Flavio D Garcia. 2015. Cryptanalysis of the Megamos Crypto automotive immobilizer. *USENIX; login* 40, 6 (2015), 17–22.

[35] Roel Verdult, Flavio D Garcia, and Josep Balasch. 2012. Gone in 360 seconds: Hijacking with Hitag2. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*. 237–252.

[36] Roel Verdult, Wei Meng, Flavio D Garcia, Dan Doozan, Baris Ege, William Enck, Alex C Snoeren, Giovanni Vigna, Tao Xie, and Nick Feamster. 2013. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *22nd USENIX Security Symposium (USENIX Security 13)*. 687–702.

[37] Juan Wang, Karim Lounis, and Mohammad Zulkernine. 2019. CSKES: A context-based secure keyless entry system. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 817–822.

[38] Pang-Chieh Wang, Ting-Wei Hou, Jung-Hsuan Wu, and Bo-Chiuan Chen. 2007. A security module for car appliances. *International Journal of World Academy Of Science, Engineering and Technology* 26, 155-160 (2007), 256.

[39] Jung-Hsuan Wu, Chien-Chuan Kung, Jhan-Hao Rao, Pang-Chieh Wang, Cheng-Liang Lin, and Ting-Wei Hou. 2008. Design of an in-vehicle anti-theft component. In *2008 Eighth International Conference on Intelligent Systems Design and Applications*, Vol. 1. IEEE, 566–569.

APPENDIX

A Cryptographic commands of F_{CC}

The description of the above (GetRV), (CompEPuK), (VerP/ValT), and (VerP/ValT) commands are listed in Table 2.

B Implementation of F_{CC} commands in P_{CC}

The implementation of cryptographic commands of F_{CC} in its realization P_{CC} is provided in Table 3.

C Proof Sketch of Theorem 3

The simulator S in this proof provides k for an uncorrupted pre-shared key from ECDH key and k' for corrupted one. This proof consists of several hybrid systems where parts of the realization are replaced with their ideal versions and the probability that an environment can distinguish these replacements is negligible, and then every preceding operation relies on its succeeding operation in the hybrid systems, which are as follows: In the first step, we define a hybrid system P^1_{CC} where all real asymmetric operations and ephemeral random values including timestamps are replaced with their ideal versions and further rely on one another. The security of this step can be reduced to the security of asymmetric operations. In the second step, we define a hybrid system P^2_{CC} where handling of private keys is replaced with the ideal version. P^2_{CC} relies on P^1_{CC} and prevents the guessing and collisions of private keys such

Table 2: Cryptographic Commands of the ideal crypto-chain functionality F_{CC}

Cryptographic Commands
Generate a fresh ephemeral random value [(GetRV)]. This command is provided to both (ID, sid, l) and (ID', sid', l'). The user (ID, sid, l) can request F_{CC} to generate an ephemeral random value. Upon receiving this request, F_{CC} forwards the request to the adversary via a restricting message. The adversary is supposed to provide an ephemeral random value $r \in \{1, ..., nr\}$ at a timestamp, say t, where nr is a large randomly selected integer. F_{CC} checks that r is fresh to prevent ephemeral random value collision. If r is not fresh, i.e., r already exists in the database, F_{CC} asks the adversary for another r until the check succeeds. Then, F_{CC} adds (ID, r, t) to the database and returns (RV, r, t) to the user.
Generate a fresh ephemeral private and compute its corresponding ephemeral public key [(CompEPuK, r, t)]. This command is provided to both (ID, sid, l) and (ID', sid', l'). The user (ID, sid, l) can request F_{CC} to compute an uncorrupted ephemeral public key Q using an ephemeral random value r and a timestamp t. Upon receiving this request, F_{CC} verifies that (ID, r, t) exists in the database and forwards this request to the adversary via a restricting message if this verification succeeds. The adversary is supposed to provide a fresh ephemeral private key $d \in \{1, ..., n\}$. F_{CC} ensures that d is fresh to prevent a private key collision. If d is fresh, i.e., d does not exist in the database, F_{CC} stores a pointer ptr_i that points to d for the user, uses the generated domain parameters (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. Otherwise, if d is not fresh, F_{CC} asks the adversary again for a new d until the check succeeds thereby preventing private key guessing.
Generate a fresh preshared key [(GenPSK, ptr_i, r, Q, r', Q')]. This command is provided to both (ID, sid, l) and (ID', sid', l'). The user (ID, sid, l) can request F_{CC} to generate a new preshared key from the pointer ptr_i pointing to the private key d, ephemeral random values r and r', and public keys Q and Q'. Upon receiving this request, F_{CC} first verifies that (r, Q) and (r', Q') are recorded in the database and then returns $(\text{PSK}, restricted)$ if any of these verifications fail. Then, F_{CC} checks whether a preshared key k has been generated by (r, Q, r', Q'), where $Q = d.G$ and returns a pointer ptr_{ii} to k if the verification succeeds. Otherwise, F_{CC} generates a new preshared key as follows. F_{CC} forwards the request to the adversary (via a restricting message $(\text{ProvidePSK}, uncorrupted, d, Q')$) to provide a new preshared key $k = F_\eta(H((d.Q'), (r', r)))$ using SHA-256 hash algorithm $H(.)$ and PRF F, where $d.Q'$ represents an ECDH key. F_{CC} ensures that k is fresh and has been generated as $k = F_\eta(H((d.Q'), (r', r)))$, adds k and (r, r') to the database, sets the pointer ptr_{ii} to k, and returns $(\text{PSKPointer}, ptr_{ii})$ to the user.
Shared secret session key derivation [(DeriveSKey, ptr_{ii}, r, r')]. This command is provided to both (ID, sid, l) and (ID', sid', l'). The user (ID, sid, l) can request F_{CC} to derive a shared secret session key k_i using k to which the pointer ptr_{ii} points to and ephemeral random values r and r'. Upon receiving this request, F_{CC} first checks whether r and r' exist in the database and returns $(\text{SKeyPointer}, restricted)$ to the user if any of the checks fails. F_{CC} derives k_i as follows: F_{CC} checks whether k_i has been derived using 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. The adversary uses $H(.)$ and a PRF F' to provide $k_i = F'_\eta(H(k.r.r'))$. 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.
Compute a cryptographic proof and a cryptographic timestamp [(CompP&E, ptr_{iv}, r, r')]. This command is provided to only (ID, sid, l). The user (ID, sid, l) can request F_{CC} to compute a cryptographic proof $proof$ and a cryptographic timestamp $ctimestamp$ using ephemeral random values r and r', and the session key k_i to which the pointer ptr_{iv} points. Upon receiving this request, F_{CC} checks whether ptr_{iv} belongs to the user and r and r' exist in the database. If these checks succeed, it provides proof by encrypting and MACing (k_i, r, r') under an AES 128-bit encryption algorithm $Enc(.)$ and 256-bit MAC algorithm $MAC(.)$ provided by the adversary. F_{CC} computes the cryptographic proof as $proof = MAC_{k_i}(x)$ at a timestamp t_3, where $x = Enc_{k_i}(k_i, r, r')$ and the cryptographic timestamp as $ctimestamp = t_3.H(k_i, r, r')$ 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 (ID', sid', l'). The user (ID', sid', l') can request F_{CC} to verify a cryptographic proof $proof$ and validate a cryptographic timestamp $ctimestamp$ using k_i to which ptr_{iv} points, ciphertext x, and a timestamp t_3. Upon receiving this request, F_{CC} first verifies that ptr_{iv} belongs to the user, and then uses a MAC verification algorithm $VMAC(.)$ and decryption of the AES 128-bit encryption algorithm $Dec(.)$ provided by the adversary to verify $proof$. F_{CC} verifies $proof$ 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) computing $t_3.H(k_i, r, r') = ctimestamp$; and ii) 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 restricted to the user.

Table 3: 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 $d \leftarrow \{1, ..., n\}$, creates a pointer ptr_i to d, uses the domain parameters (p, a, b, G, n, h) 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 (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 $VMAC(proof, x) = 1$ (at a timestamp t_6) 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.

that any distinguishing environment on P^2_{CC} is reduced to the DDH assumption. In the third step, we define another hybrid system P^3_{CC} where real preshared key generation is replaced with an ideal version. P^3_{CC} relies on P^2_{CC} and P^3_{CC} does not prevent the guessing or collisions of preshared keys, which are provided by the simulator as presented above. In step four, we define a hybrid argument P^4_{CC} where real symmetric encryption and shared secret session key derivation operations are replaced with their ideal versions and guessing and collisions of keys are prevented. Besides, P^4_{CC} relies on P^3_{CC}. The security of this step can be reduced to the security of encryption and key derivation schemes. In step five, we replace real MACs with their ideal versions using a hybrid system P^5_{CC}. While P^5_{CC} relies on P^4_{CC}, the security of this step can be reduced to the security of the MAC scheme. We combine all the five steps above and deduce that the simulator is responsive. This concludes the proof sketch of this theorem.

D Theorem and Proof of *MKD*

The following theorem states that the *MKD* is a secure universally composable relay resilience mutual authentication and key derivation protocol.

Theorem 4. *Let $M_{A(MKD)}$ and $M_{B(MKD)}$ be the machines modelling the MKD as described above, let c be a sequence of cryptographic operations of P_{CC} and F_{CC} such that every operation c_i in c always rely on the preceding operation c_{i-1}, and let F_{MKD} be the ideal functionality for mutual authentication and key derivation with parameter $s_{key} = authenc - key$. Then, the following holds true:*

$$M_{A(MKD)}|M_{B(MKD)}|F_CC[c] \leq_R F_{MKD}$$

Proof: We say that a user ID is corrupted if its preshared key is corrupted. An instance (ID, sid, l) is corrupted if it outputs corrupted when asked for its corruption status by the environment. An instance is explicitly corrupted if its control is taken over by the adversary. We define a responsive simulator S and show that $E|M_{A(MKD)}|M_{B(MKD)}|F_{CC}[c] \equiv E|S|F_{MKD}$ for all environments $E \in Env_R(M_{A(MKD)}|M_{B(MKD)}|F_{CC}[c])$. Note that S fulfils the runtime conditions and immediately responds to restricting messages as long as the environment E does the same with overwhelming probability. S internally simulates the protocol $M_{A(MKD)}|M_{B(MKD)}|F_{CC}[c]$, uses F_{MKD} to keep the corruption statuses of user instances, synchronizes the simulated instances of $M_{A(MKD)}/M_{B(MKD)}$. To model the protocol, S needs to first initialize F_{CC} by sending a message to F_{CC}. In response, S receives domain parameters (p, a, b, G, n, h) and cryptographic operations c and then uses the parameters and operations to simulate F_{CC}. Then, S requests the environment E for the cryptographic algorithms required in the execution of F_{CC}.

If F_{MKD} indicates that a user (ID, sid, l) has started a mutual authentication and key derivation based on $c_i(c_{i-1})$, S updates its internal simulation accordingly. If an ephemeral random value r_B and public key $Q_B = d_B.G$ are accepted by an uncorrupted initiator instance (ID_A, sid_A, A) and the initiator outputs a pointer ptr_{iv} to a shared secret session key k_i, then S instructs F_{MKD} to create a session from (ID_A, sid_A, A) and the responder instance (ID_B, sid_B, B) that created encryption and MAC in the second message of the protocol. Then, F_{CC} will ask S to provide the session key value and S provides and uses the same value in its simulation. Finally, S instructs F_{MKD} to output the session key pointer for (ID_A, sid_A, A). If S receives a message that some instance has closed its session, S updates its internal simulation accordingly and returns *Okay*. To process messages for/from corrupted instances, S uses its internal

simulation. Due to the use of restricting messages, we say that all operations performed by F_{CC} are always successful and have no side effects on $M_{A(MKD)}$ or $M_{B(MKD)}$.

Furthermore, we argue that the simulation is perfect in the case of an honest initiator instances during mutual authentication and key derivation and in the case of an honest responder instances during mutual authentication and key derivation. Firstly, let (ID_A, sid_A, A) be an honest instance of M_A that wants to establish a mutual authentication and key derivation session with user $ID^{'}$. The instance will use F_{CC} to generate ephemeral random values, encrypt and decrypt messages, create and verify MACs, generate preshared keys via ECDH keys, and derive shared secret session keys, and compute cryptographic proof and cryptographic timestamp. We need to argue that S finds an honest responder instance that can be paired with (ID_A, sid_A, A) and further ensure relay resilience: If a session key pointer is output by (ID_A, sid_A, A) within the maximum computation time of MKD, this shows that (ID_A, sid_A, A) must have accepted the second message and validated the expected time of completing the MKD, and the derived key must still be uncorrupted. Otherwise, the protocol would stop and block according to our corruption model. Thus, there exists some instance of $ID^{'}$, say $(ID^{'}, sid^{'}, l^{'})$, that provided the cryptographic proof $proof = MAC_{k_i}(x)$ and $x = Enc_{k_i}(k_i, r_A, r_B)$, where $k_i = F^{'}_{\eta}(H(k.r_A.r_B))$, $k = F_{\eta}(H((d_A.Q_B), (r_B, r_A)))$, $Q_B = d_B.G$, timestamp t_3 and cryptographic timestamp $ctimestamp = t_3.H(k_i.r_A.r_B)$. Note that d_A is the private key of (ID_A, sid_A, A) and d_B is the private key of $(ID^{'}, sid^{'}, l^{'})$. This shows that $(ID^{'}, sid^{'}, l^{'})$ is uncorrupted. As $ID^{'}$ is uncorrupted, the instance cannot be explicitly corrupted by the adversary. We now argue that $(ID^{'}, sid^{'}, l^{'})$ is a responder, i.e., $l^{'} = B$: If $l^{'} = A$, then z would imply that $(ID^{'}, sid^{'}, l^{'})$ received and accepted the first message of the protocol (Q_A, r_A, t_1) provided by an uncorrupted instance of ID_A, where $Q_A = d_A.G$ and d_A is the private key of the instance ID_A. However, as d_A/Q_A is ideally computed and cryptographic operations are integrated, there is only one instance that would provide and send such a message, namely (ID_A, sid_A, A), which does not output any encryption or MAC before accepting the first message. This implies $l^{'} = B$. We observe that as d_A/Q_A and d_B/Q_B are ideally computed and validated, the session key, cryptographic proof, and cryptographic timestamp derived from them will be considered uncorrupted in F_{CC} and only (ID_A, sid_A, A) and $(ID^{'}, sid^{'}, l^{'})$ can get a pointer to the session key and know the cryptographic proof and cryptographic timestamp which corresponds to the behavior of F_{MKD}.

Lastly, let (ID_B, sid_B, B) be an uncorrupted instance of $M_{B(DKD)}$ that wants to establish a mutual authentication and key derivation session with $ID^{'}$. We need to show that (ID_B, sid_B, B) is already part of a session in F_{AKD} when it outputs a session key pointer and every operation and action up to that point can be perfectly simulated. We observe that if such a pointer is output by (ID_B, sid_B, B) within the maximum computation time of MKD, this shows that it has accepted the first message of the protocol and $ID^{'}$ must still be uncorrupted. This shows that there is an instance of $ID^{'}$, say $(ID^{'}, sid^{'}, l^{'})$, that provided and sent the message (Q_A, r_A, t_A), where $Q_A = d_A.G$, where d_A is the private key of $(ID^{'}, sid^{'}, l^{'})$. Thus, this instance is uncorrupted. We now argue that this instance is an initiator, i.e., $l^{'} = I$: Suppose $l^{'} = R$, the instance $(ID^{'}, sid^{'}, l^{'})$ cannot have received its first message of the protocol before (ID_B, sid_B, B) has received its first message of the protocol. Thus, $l^{'} = I$. We argue that as $(ID^{'}, sid^{'}, l^{'})$ has derived a session key and cryptographic proof via its accepted second message of the protocol, it has completed its part of the mutual authentication and key derivation. Furthermore, the instance (ID_B, sid_B, B) is the only instance that can encrypt and MAC proof in the second message of the protocol as d_B/Q_B is unique and ideal computed and cryptographic operations are integrated, thus $(ID^{'}, sid^{'}, l^{'})$ is in a session with (ID_B, sid_B, B). Note that as both instances use the same uncorrupted private keys d_A and d_B to derive a session key and cryptographic proof and $d_A \neq d_B$, these instances will output pointers to the same uncorrupted session key and cryptographic proof. Thus, the simulation is perfect in this case.

We note that other cases such as honest instances after mutual authentication and key derivation as well as corrupted instances are omitted due to page limit.

By Theorem 3, we can now replace $F_{CC}[c]$ by its realization $P_{CC}[c]$ which yields that the MKD when using the actual cryptographic operations is a universally composable relay resilience mutual authentication and key derivation protocol.

Proposition 1. *Let $M_{A(MKD)}$ and $M_{B(MKD)}$ be machines as defined above, let $F_{CC}[c]$ and $P_{CC}[c]$ be as in Theorem 3 and M^* enforces well-behaved environments. Then, the following holds true:*

$$M^*|M_{A(MKD)}|M_{B(MKD)}|P_{CC}[c] \leq_R M^*|F_{MKD}|F_{CC}[c]$$

Proof: This proposition follows easily from Theorem 2, Theorem 3, and Theorem 4 that the machines $M_{A(MKD)}$ and $M_{B(MKD)}$ constitute a well-behaved environment when combined with M^* and any other environment E.

E Photograph of Crypto-Chain Implementation

Figure 5 presents the photography of Crypto-Chain implementation in our laboratory. This figure shows the components that are used in our implementation.

F Theorem and Proof Sketch of the Enhanced Megamos Crypto

The following theorem states that the *Enhanced Megamos Crypto* is a secure universally composable relay resilience mutual authentication and key derivation protocol.

Theorem 5. *Let $M_{A(MC)}$ and $M_{B(MC)}$ be the machines modeling the Enhanced Megamos Crypto as described above. Let c be a sequence of cryptographic operations of P_{CC} and F_{CC} such that every operation c_i in c always rely on the preceding operation c_{i-1}, and let F_{MKD} be the ideal functionality for mutual authentication and key derivation with parameter $s_{key} = authenc - key$. Then, the following holds true:*

$$M_{A(MC)}|M_{B(MC)}|F_{CC}[c] \leq_R F_{MKD}$$

Proof Sketch: The proof sketch of this theorem is straightforward and does not require any probabilistic reasoning. The simulator S in this proof internally simulates the protocol $M_{A(MC)}|M_{B(MC)}|F_{CC}[c]$, uses instances $M_{A(MC)}$ and $M_{B(MC)}$, and show that

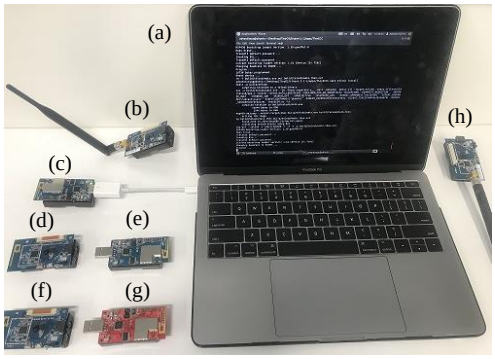

Figure 5: Photograph of Crypto-Chain Implementation in our laboratory. Notations: (a) MacBook Pro, (b)/(h) CM3000 sensor, (c)/(e) CM5000 sensor, (d)/(f) CM4000 sensor, (g) XM1000 sensor.

$E|M_{A(MC)}|M_{B(MC)}|F_{CC}[c] \equiv E|S|F_{MKD}$ for all environments $E \in Env_R(M_{A(MC)}|M_{B(MC)}|F_{CC}[c])$. Just as the *MKD*, by Theorem 3, we can replace $F_{CC}[c]$ with its realization $P_{CC}[c]$ which yields that the *Enhanced Megamos Crypto* is a secure universally composable relay resilience mutual authentication and key derivation protocol.

Proposition 2. *Let $M_{A(MC)}$ and $M_{B(MC)}$ be machines as defined above, let $F_{CC}[c]$ and $P_{CC}[c]$ be as in Theorem 3 and M^* enforces well-behaved environments. Then, the following holds true:*

$$M^*|M_{A(MC)}|M_{B(MC)}|P_{CC}[c] \leq_R M^*|F_{MKD}|F_{CC}[c]$$

Proof: It can be seen easily from Theorem 2, Theorem 3, and Theorem 5 that $M^*|M_{A(MC)}|M_{B(MC)}$ constitutes a well-behaved environment when combined with E.

G Theorem and Proof Sketch of the Enhanced Hitag-AES/Pro

The following theorem states that the *Enhanced Hitag-AES/Pro* is a secure universally composable relay resilience mutual authentication and key derivation protocol.

Theorem 6. *Let $M_{A(HP)}$ and $M_{B(HP)}$ be the machines modelling the Enhanced Hitag-AES/Pro that encrypt and MAC messages, decrypts and VMAC messages, provide/verify cryptographic proof, and verifies expected time of completing the protocol execution. Let c be a sequence of cryptographic operations of P_{CC} and F_{CC} such that every operation c_i in c always rely on the preceding operation c_{i-1}, and let F_{MKD} be the ideal functionality for mutual authentication and key derivation with parameter $s_{key} = authenc - key$. Then, the following holds true:*

$$M_{A(HP)}|M_{B(HP)}|F_{CC}[c] \leq_R F_{MKD}$$

The proof of this theorem is similar to the MKD and Megamos Crypto, except that, S internally simulates the protocol $M_{A(HP)}|M_{B(HP)}|F_{CC}[c]$, synchronizes the simulated instances of $M_{A(HP)}/M_{B(HP)}$, and keeps the corruption statuses of user instances using F_{MKD}. S shows that $E|M_{A(HP)}|M_{B(HP)}|F_{CC}[c] \equiv E|S|F_{MKD}$ for all environments $E \in Env_R(M_{A(HP)}|M_{B(HP)}|F_{CC}[c])$. Just as in the MKD and Megamos Crypto, $F_{CC}[c]$ can be replaced with $P_{CC}[c]$ which yields that the *Enhanced Hitag-AES/Pro* is a secure universally composable relay resilience mutual authentication and key derivation protocol.

Proposition 3. *Let $M_{A(HP)}$ and $M_{B(HP)}$ be machines as defined above, let $F_{CC}[c]$ and $P_{CC}[c]$ be as in Theorem 3 and M^* enforces well-behaved environments. Then, the following holds true:*

$$M^*|M_{A(HP)}|M_{B(HP)}|P_{CC}[c] \leq_R M^*|F_{MKD}|F_{CC}[c]$$

Proof: Using Theorem 2, Theorem 3, and Theorem 7, it can be seen easily that $M^*|M_{A(HP)}|M_{B(HP)}$ creates a well-behaved environment with any environment E.