MVSec: Secure and Easy-to-Use Pairing of Mobile Devices with Vehicles

Jun Han, Yue-Hsun Lin, Adrian Perrig, Fan Bai

May 26, 2014

CMU-CyLab-14-006

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

MVSec: Secure and Easy-to-Use Pairing of Mobile Devices with Vehicles

Jun Han[†‡], Yue-Hsun Lin[†], Adrian Perrig[‡], Fan Bai[§]
[†]Carnegie Mellon University {junhan, tenma}@cmu.edu
[‡]ETH Zurich adrian.perrig@inf.ethz.ch
[§]General Motor Research fan.bai@gm.com

ABSTRACT

With the increasing popularity of mobile devices, drivers and passengers will naturally want to connect their devices to their cars. Malicious entities can and likely will try to attack such systems in order to compromise other vehicular components or eavesdrop on privacy-sensitive information. It is imperative, therefore, to address security concerns from the onset of these technologies. While guaranteeing secure wireless vehicle-to-mobile communication is crucial to the successful integration of mobile devices in vehicular environments, usability is of equally critical importance. Several researchers proposed different methods for key agreement between two devices that share no prior secret. However, many of these proposals do not take advantage of the vehicular environment. With *MVSec*, we propose several novel approaches to secure vehicle-to-mobile communication tailored specifically for vehicular environments. We present novel security protocols and their security analysis. We also provide complete implementation and user study results demonstrating the feasibility and the usability of MVSec.

1. INTRODUCTION

With the proliferation of wireless devices using Wi-Fi and Bluetooth technologies, security of their communication is a vital concern as numerous real-world attacks have been reported [1, 2, 3]. Insecure wireless communication may allow attackers to easily eavesdrop or launch Man-in-the-Middle (MitM) attacks, impersonating legitimate communicating devices.

Efforts to eradicate such attacks have inspired many research proposals as well as industrial solutions, namely to provide secure pairing between the devices by "bonding" them to establish a secure channel. However, it is still significantly difficult for human users to easily determine which devices are being paired because of the invisible nature of wireless communication. Hence, researchers propose demonstrative identification, which affirms to the human user which devices are actually communicating leveraging out-of-band channels. [4].

However, many naive solutions attempting to establish such secure pairing for any two devices introduces a trade-off. In many cases, increasing security leads to decreased usability for users, which becomes a significant hindrance for wide adoption of the technology by the general public. On the other hand, decreased usability may cause a security breach in these protocols. This is best exemplified by the use case scenario when a user tries to pair her phone with a friend's phone using Bluetooth. The state-of-the-art solutions require the user to either copy a passkey displayed on one device to the other, compare two passkeys displayed on both devices, or to enter a hard-to-guess passkey on both devices. However, the security of such protocols often rely on the passkey not being repeated or easy-to-guess, requiring the users to input hard-to-guess passkeys to guarantee the protocol security [5, 6]. These designs, however, lead to multiple problems in practice. Many devices actually display a repeated and/or easy-to-guess passkeys (e.g., 000000, 123456, etc.) [7, 8]. Also, many users tend to make fatal mistake of inputting easy-to-guess passkeys [9].

In this paper, we delve into a specific problem of vehicular environments. The proliferation of smartphones coupled with emerging smarter vehicles allows constant exchange of sensitive information over wireless communication. For example, different automotive manufacturers and smartphone companies established Car Connectivity Consortium (CCC) and have formed *Mirror Link*, a standard for integrating smartphones and the vehicles to enable access to the phones using car's control, display, and speakers [10]. Apple has recently revealed *CarPlay*, its first in-car infotainment system which allows to an integration of iOS devices with vehicles [11, 12]. There are already 18 car manufacturers that are partnering with Apple in this project. Current version of *CarPlay* only supports a USB connection from the vehicle to iOS devices, but we foresee a high potential for transition to wireless technology. In addition, Ford launched *MyFord Touch*, which allows a car to act as a Wi-Fi hotspot for passengers with Wi-Fi enabled devices [13]. In addition to pairing with personal cars, we expect more frequent pairing uses cases for widely used rental car services – both traditional and short-term rental cars (e.g., Zipcar and Mobility) [14, 15]. The recent increasing popularity of Lyft, a peer-to-peer ride-sharing services in California is yet another potential candidate [16, 17].

Unfortunately, coupling of smartphones and vehicles introduces a new avenue of potential attacks if the wireless channel is not secured. Such attacks on wireless communication in cars have already been demonstrated by the project

named *Car Whisperer* [18, 19]. The researchers show that attackers in adjacent cars (within the Bluetooth transmission range) exploit the vulnerability of Bluetooth pairing to eavesdrop on the conversations that take place in the victim's car. Although launching such attacks may not seem simple and plausible at a first glance, they are certainly within the realm of possibility especially for high-value targets (e.g., celebrities, politicians, CEOs, and etc.) that provide more incentives for the attackers. Furthermore, such targets are more likely to drive luxury vehicles that embrace next-generation vehicle-to-mobile convergence systems. Do current cars effectively protect themselves from remote attackers attempting to compromise vehicular components? Can we be convinced that the sensitive information in our vehicles is not being maliciously transmitted to attackers in other nearby cars on the road, in parking lots, or gas stations connected via Bluetooth or Wi-Fi?

To address these problems, we present *MVSec*, the first secure key agreement scheme tailored specifically for vehicular environments, providing strong security guarantees and easy usability. MVSec leverages out-of-band channels such as sound or light as its communication medium because commodity hardware such as LED, ambient light sensor, speaker, and microphone are readily available in cars and/or smartphones. MVSec allows a user, typically the driver, to simply press a button on each device (the car and the phone) to initiate the protocols. For the protocols that leverage sound, all the user needs to do is to simply verify that both the car and the intended mobile device emit a short beep. Similarly, for protocols that leverage light emission, the user simply needs to place the mobile device in the glove compartment for a short amount of time. We present detailed explanation of more protocols and their security analysis in Section 4.

This paper makes the following **contributions**. We provide (1) a description of MVSec vehicle-to-mobile pairing protocols which leverages different cryptographic schemes based on various out-of-band channels readily available in commercial vehicles and mobile devices; (2) the formal security analysis of these protocols; (3) an implementation of the MVSec protocols on Android smartphones; and (4) experimental results to demonstrate the usability of MVSec, as it requires minimal user involvement.

2. PROBLEM DEFINITION

This section presents the goals we plan to achieve given the constraints, lists the assumptions we hold, and discusses the attacker model.

The main goal of MVSec is to present a complete system that provides a secure and usable communication between the car and the smartphone. If an attacker is present and launches an attack, it will be clear to the user that an error has occurred, so that the user can immediately abort the pairing process. The main properties MVSec tries to achieve are the following. MVSec achieves **secrecy** by allowing the driver's phone and the car to hide information from unintended devices. It also achieves **authenticity** and **integrity** by allowing the driver's phone and the car to validate that unaltered data arrived from the claimed sender. MVSec also achieves **demonstrative identification** by enabling the user to explicitly be aware of which devices are actually communicating via the wireless communication.

We also categorize some of the constraints that pose challenges in achieving the aforementioned goals. The phone and the car initially do not share any prior secret, nor depend on any Trusted Third Party (TTP) for exchanging the secret. In addition, MVSec incurs minimal hardware cost by leveraging available hardware commonly installed in today's cars and smartphones to communicate via out-of-band (OOB) channels (discussed further in Section 4).

2.1 Assumptions

Given the specific vehicular setting, we make the following assumptions to achieve the aforementioned goals. We assume that the OOB channel *does not require user diligence*. This is a necessary assumption to ensure high usability. We also assume that there is *no malware* on the vehicle or mobile device. If there is malicious code on the mobile device, a secure pairing protocol cannot establish a secret because attackers can gain the shared secret through the malware.

2.2 Adversary Model

This section presents the attacker model by describing the attacker goals and capabilities.

Attacker Goals The goals of the attacker is to break the security properties that MVSec aims to achieve, namely to break secrecy and authenticity of vehicle-to-mobile communication.

Attacker Capabilities We assume that the attacker can perform both passive and active attacks. A passive attacker can perform attacks without actively participating in the protocol, such as eavesdropping. An active attacker follows a Dolev-Yao attacker model who are able to perform various types of attacks in addition to eavesdropping – data injection attacks, denial-of-service, man-in-the-middle (MitM), etc. In this paper, we concentrate on defending against the MitM attack.

There are two types of attackers that we envision in a secure pairing environment with vehicle and mobile devices. Different attackers are limited by different capabilities given corresponding constraints.

(1) Attacker inside the vehicle. In this case, an attacker is present inside the vehicle. This type of attacker can launch powerful attacks because the attacker has access to the visual display in the vehicle. Computational power is not a limitation for this type of attacker because he can have remote access to powerful computational devices. The only limitation would be the bandwidth of the network connectivity.

(2) Attacker is outside of the vehicle but within wireless transmission range (<100 meters away). Attackers who are outside of the car can still perform powerful attacks. This type of attacker is constrained in many aspects compared to the first type as the attacker cannot see the visual display and listen to sound emissions from the vehicle and the mobile devices, and cannot launch out-of-band channel attacks. The attacker can still perform considerable attacks such as jamming, eavesdropping, and man-in-the-middle attacks, especially if equipped with powerful devices such as a high-gain antenna.

3. RELATED WORK

Many researchers have investigated the problem of securely pairing two devices that do not share prior secret key, or have each other's authentic public key. One of the main challenges in secure pairing, however, is to provide usability

while guaranteeing security.

Commodity wireless solutions such as Bluetooth or Wi-Fi have standards that attempt to provide a secure exchange of credentials (e.g., Bluetooth Secure Simple Pairing (SSP) [6] and Wi-Fi Protected Setup [20]). We illustrate as an example the details of one of the SSP protocols called *numeric comparison*. This protocol consists of two phases in performing a secure pairing. In the first phase, a pair of devices exchange their public keys (e.g., Diffie Hellman Key Exchange). Then in the second phase, both devices perform verification on the received public keys by requiring the user to verify if the displayed numbers on both devices are identical. Once the user performs a successful verification, the devices then establish a secure connection. However, Kuo et al. [5] highlight that large attack surfaces for these specifications exist, and provide recommendations to improve usability. For example, the security of the *numeric comparison* method depends on the displayed number to be hard to guess, and not repeated. However, in many products, manufacturers are not careful in implementing the standards, and cause potential security vulnerabilities.

Many researchers propose different solutions to achieve secure pairing while preserving usability. They attempt to leverage different types of OOB channels to provide demonstrative identification. First, researchers propose using visual channels as the OOB channel. McCune et al. propose Seeing-is-Believing (SiB) [21], a solution that allows two smartphones to securely exchange each other's public keys using QR codes and phone cameras. SiB, however, is not well suited for a vehicular setting because it requires extra hardware such as cameras, which is not present in vehicles. SiB also requires user diligence as the users need to actively take pictures of the QR code.

The audio channel has also been leveraged by researchers to be used as another OOB channel. HAPADEP [22] first exchanges the public keys of two devices via sound, using built-in speakers and microphones. In the verification phase, the two devices reveal data commitments by emitting another sequence of sound signals. The user is required to verify if the two sound signals are identical. However, this comparison is quite cumbersome for the user, and leads to a high error rate and seriously limits usability.

Kuo et al. propose Message-In-a-Bottle (MIB) [23], which leverages a Faraday cage, a special hardware device that ensures authenticity and secrecy of the communication inside the cage. The cage prevents any wireless signals communicated between the nodes inside to be leaked to attackers outside the cage. Also, attackers are infeasible to inject wireless signals into the cage. While this may be applicable in a sensor network context, we cannot assume that such cage to be installed in a vehicle.

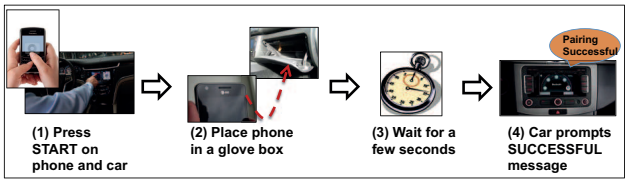
There are solutions that explicitly require human users to play the role of the OOB channel. Gehrmann et al. [24] propose a series of protocols named Manual Authentication (MANA) by allowing the user to read and input short strings to different devices. Although MANA protocols may seem secure, we claim that MANA is not suitable for solving our problem because it requires user diligence which often leads to human errors, eventually resulting in possible attacks.

Many researchers also investigate the usability of different secure pairing proposals [25, 26, 27]. The results of these studies conclude that authentication mechanisms that involve active user participation (e.g., comparing numbers) is one of the most important factor in influencing user's perception of the convenience of a specific pairing solution.

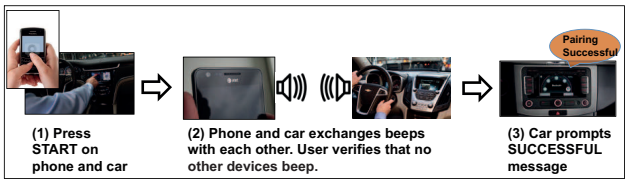
4. MVSec PROTOCOLS

This section presents the overview of the MVSec protocols at a high-level, discusses the OOB channel selection, and then delves into the protocol details. The main goal of MVSec is to allow a user to securely pair his smartphone with his vehicle such that an attacker will not successfully launch MitM attacks.

To achieve this goal, we first need to overcome the challenge of providing *demonstrative identification*, to ensure that the vehicle and the intended smartphone are in fact communicating with each other. We leverage out-of-band (OOB) communication channels as a solution. Different from the primary channels used by the devices, e.g., Wi-Fi or Bluetooth, an OOB channel is a separate communication medium between the communicating devices (e.g, humans, light, sound, vibrations, etc.).



(a) Strong OOB Channel (light)



(b) Weak OOB Channel (sound)

Figure 1: Overview of MVSec using light and sound as OOB channels

4.1 Out-of-band Channel Selection

MVSec leverages two types of OOB channels for different protocols described in detail in Section 4.2. According their characteristics, they are categorized into *strong* and *weak* OOB channels.

Strong OOB Channel. A strong OOB channel guarantees both *secrecy* and *authenticity*. We select **light** in a vehicle's closed glove compartment as the strong OOB channel because it provides both of these security properties. We assume that the glove compartment does not leak any light signal, thus provides secrecy. This channel also provides authenticity because only the vehicle will emit light signals. This is because no other device is inside the compartment as the driver first verifies that other devices are not placed inside the compartment during protocol execution.

In addition to considering the security properties, we choose light to conform to the assumptions made in Section 2.1. The OOB channel needs to (1) be readily available in vehicles and smartphones today in order to be easily deployable, and (2) provide high usability, i.e., the OOB channel needs to have a relatively fast data rate and be easy and pleasant

to use (i.e., should not require user diligence nor annoy the users). We define relatively fast data rate to be faster than the OOB channel used as baseline case, which is manual human input (explained further in Section 6). This OOB channel allows such usability because the only task that the user performs is to press a button on both the vehicle and the smartphone, and place the smartphone inside the glove compartment. After waiting for a few seconds, during which the vehicle transmits signals via blinking lights to the smartphone, the pairing process successfully completes. The steps are depicted in Figure 1(a).

Weak OOB Channel. A weak OOB channel provides only *authenticity*. We select **sound** signals as the weak OOB channel. This channel provides authenticity because we assume that it is easy for the user to identify that the sound beeps are originating only from the intended devices (e.g., vehicle and driver's smartphone). If an unintended device beeps, the user simply aborts the protocol. We assume that the beeps are sufficiently long and loud enough for the user to easily identify the origin of the beeps. We assert that this is a realistic assumption, because smartphone users generally distinguish who's phone is ringing when (s)he hears a phone ring. We also use sound signals because of the ubiquitous deployments of microphones and speakers in vehicles and smartphones. Figure 1 depicts the steps the user performs.

We provide different variation of protocols using different cryptographic primitives that leverage the two aforementioned OOB channels. The protocols are described in detail in the next section.

Table 1: Notations for MVSec protocols

Notation	Description
$\overset{Light}{\Longrightarrow}$	Strong OOB channel using light in a closed glove compartment
$\overset{Sound}{\Longrightarrow}$	Weak OOB channel using sound
$\overset{BT}{\longrightarrow}$	In-band Bluetooth channel
$M_K(x)$	MAC (e.g., HMAC) computed over the input x, using key K
g^x	Diffie-Hellman public parameter (omitted *mod* p for clarity)
$H(x)$	Cryptographic hash (e.g., SHA-3) of input x
$\{x\}_K$	Symmetric encryption (e.g., AES) of input x using key K
$[x]_i$	Truncated i bit string of input x
$\{0,1\}^i$	Random binary string with length i

4.2 MVSec Protocol Details

This section describes three MVSec protocols that leverage light and sound signal in a closed compartment as the OOB channel. We present the underlying cryptographic primitives of these protocols and compare them to select the most efficient protocol. Table 1 denotes the notations used in the MVSec protocols.

4.2.1 Strong OOB Channel

We now describe protocol details leveraging the light as an OOB channel. This protocol makes use of the Encrypted Key Exchange (EKE) [28] and is depicted in Figure 2.

MVSec-I: Protocol leveraging Encrypted Key Exchange (EKE). A conventional EKE scheme allows two participating entities to use a shared low-entropy password to derive a temporary shared key that can be used to authenticate the key exchange messages. We use a variant of the EKE scheme by treating a short shared secret K_s (20 bits) as a low entropy password. K_s is first transmitted via the light signal in Step 2. In Steps 3 and 4, both the vehicle and the smartphone transmit their DH public keys encrypted with K_s. Then the vehicle and the mobile device also performs key confirmation in Steps 4 and 5.

MVSec-I: Protocol using EKE

1. $User$: Presses start buttons on A and B. Places B in the glove compartment.
2. A : $K_s \xleftarrow{R} \{1,0\}^\ell$
   $A \overset{Light}{\Longrightarrow} B$: K_s
3. $A \overset{BT}{\longrightarrow} B$: $\{g^a\}_{K_s}$
   B : Decrypts $\{g^a\}_{K_s}$ to get g^a; Computes shared key $K' = (g^a)^b$.
4. $B \overset{BT}{\longrightarrow} A$: $\{g^b\}_{K_s} || M_{K'}(n_A)$ where $n_A = H(\{g^a\}_{K_s})$
   A : Decrypts $\{g^b\}_{K_s}$ to get g^b; Computes shared key $K = (g^b)^a$; $M_{K'}(n_A) \overset{?}{=} M_K(H(\{g^a\}_{K_s}))$; aborts if verification failed.
5. $A \overset{BT}{\longrightarrow} B$: $M_{K'}(n_B)$ where $n_B = H(\{g^b\}_{K_s})$
   B : $M_{K'}(n_B) \overset{?}{=} M_K(H(\{g^b\}_{K_s}))$; aborts if verification failed.

Figure 2: Key agreement protocol between a vehicle (A) and a smartphone (B) using EKE with light as the strong OOB channel. In practice, $\ell = 20$.

4.2.2 Weak OOB Channel

MVSec leverages *Short Authenticated Strings (SAS)* [29, 30, 31] which uses commitment/decommitment schemes prior to transmitting the short hash comparisons for verification. This approach is preferred over a naive approach of sending short hash values over the weak OOB channel for verification. The reason is that attackers may be able to launch attacks to find hash collisions.

MVSec-II: SAS with random nonce. Figure 3 presents first of the two protocols that leverages weak OOB channel. This protocol uses short authenticated strings (SAS) with random nonces.

In Steps 2 and 3, the vehicle and the mobile device generate and exchange commitments. The commitments are the hashes of each party's DH public key concatenated with its random nonce, n_A and n_B. Steps 4 and 5 allows the vehicle and the mobile device to transmit the decommitment in order to verify public keys with stored commitments. Once both parties verify the hash by comparisons, they will transmit the SAS messages mutually through the weak OOB channel in Steps 6 and 7. Note that the vehicle can also use light in a glove compartment to transmit this SAS message, although the secrecy is not needed in this protocol. Meanwhile, the vehicle and the smartphone both verify the received SAS value is equal to the SAS value they sent out. If two SAS values match, both devices generate their individual DH ephemeral key, K and K'. Such as the previous protocol, the key confirmation procedure would perform in Steps 8 – 11.

MVSec-III: SAS with bit truncation. We now describe the second protocol leveraging the weak OOB channel in Figure 4. The vehicle and the smartphone transmit their

MVSec-II: Protocol using SAS with Nonce

1. $User$: Presses start buttons on A and B. Aborts if devices other than A or B beep during execution.
2. A : Nonce $n_A \xleftarrow{R} \{1,0\}^{\ell}$.
   $A \xrightarrow{BT} B$: $C_A = H(g^a || n_A)$.
3. B : Nonce $n_B \xleftarrow{R} \{1,0\}^{\ell}$.
   $B \xrightarrow{BT} A$: $C_B = H(g^b || n_B)$.
4. $A \xrightarrow{BT} B$: $g^a || n_A$.
   B : $C_A \stackrel{?}{=} H(g^a || n_A)$; aborts if verification fails.
5. $B \xrightarrow{BT} A$: $g^b || n_B$.
   A : $C_B \stackrel{?}{=} H(g^b || n_B)$; aborts if verification fails.
6. $A \overset{Sound}{\Longrightarrow} B$: $SAS_A = n_A \oplus n_B$.
   B : $SAS_B = n_B \oplus n_A$
   $SAS_A \stackrel{?}{=} SAS_B$; aborts if verification fails.
   Computes shared key $K = (g^a)^b$;
7. $B \overset{Sound}{\Longrightarrow} A$: SAS_B.
   A : $SAS_B \stackrel{?}{=} SAS_A$; aborts if verification fails.
   Computes shared key $K' = (g^b)^a$.

Key confirmation (check $K' \stackrel{?}{=} K$)

8. A : Nonce $n'_A \xleftarrow{R} \{1,0\}^{\eta}$.
   $A \xrightarrow{BT} B$: $n'_A || M_{K'}(n'_A)$
9. B : Nonce $n'_B \xleftarrow{R} \{1,0\}^{\eta}$.
   $B \xrightarrow{BT} A$: $n'_B || M_K(n'_A || n'_B)$
10. A : $M_K(n'_A || n'_B) \stackrel{?}{=} M_{K'}(n'_A || n'_B)$; abort if confirmation fails.
    $A \xrightarrow{BT} B$: $M_{K'}(n'_B)$
11. B : $M_{K'}(n'_B) \stackrel{?}{=} M_K(n'_B)$; abort if confirmation fails.

Figure 3: Key agreement protocols between a vehicle (A) and a smartphone (B) using SAS with random nonce, leveraging weak OOB channel providing only authenticity. In practice, $\ell = 20$ and $\eta = 256$ (HMAC-SHA3).

MVSec-III: Protocol using SAS with Hash

1. $User$: Presses start buttons on A and B. Aborts if devices other than A or B beep during execution.
2. $A \xrightarrow{BT} B$: $C_A = H(g^a)$.
3. $B \xrightarrow{BT} A$: $C_B = H(g^b)$.
4. $A \xrightarrow{BT} B$: g^a
5. B : $C_A \stackrel{?}{=} H(g^a)$ verifies g^a and abort if verification fails. Computes shared key $K = (g^a)^b$.
   $B \xrightarrow{BT} A$: g^b
6. A : $C_B \stackrel{?}{=} H(g^b)$ verifies g^b and abort if verification fails. Computes shared key $K' = (g^b)^a$.
7. $A \overset{Sound}{\Longrightarrow} B$: $SAS_A = [H(K')]_{\ell}$.
   B : $SAS_B = [H(K)]_{\ell}$;
   $SAS_A \stackrel{?}{=} SAS_B$; aborts if verification fails.
8. $B \overset{Sound}{\Longrightarrow} A$: SAS_B.
   A : $SAS_B \stackrel{?}{=} SAS_A$; aborts if verification fails.

Key confirmation (check $K' \stackrel{?}{=} K$)

9. A : $n'_A \xleftarrow{R} \{1,0\}^{\eta}$.
   $A \xrightarrow{BT} B$: $n'_A || M_{K'}(n'_A)$
10. B : $n'_B \xleftarrow{R} \{1,0\}^{\eta}$.
    $B \xrightarrow{BT} A$: $n'_B || M_K(n'_A || n'_B)$
11. A : $M_K(n'_A || n'_B) \stackrel{?}{=} M_{K'}(n'_A || n'_B)$; abort if confirmation fails.
    $A \xrightarrow{BT} B$: $M_{K'}(n'_B)$
12. B : $M_{K'}(n'_B) \stackrel{?}{=} M_K(n'_B)$; abort if confirmation fails.

Figure 4: Key agreement protocols between a vehicle (A) and a smartphone (B) using SAS with truncated bits, leveraging weak OOB channel providing only authenticity. In practice, $\ell = 20$ and $\eta = 256$ (HMAC-SHA3).

commitment messages in Steps 2 and 3. These commitments are hash of their DH public keys. Like MVSec-II, they reveal the public keys to each other in Steps 4 and 5. After verifying the public keys by comparing the hashes, both parties generate negotiated DH key K and K' in Steps 5 and 6. To confirm the correctness of negotiated DH key, two parties exchange the SAS messages in Steps 7 and 8 via the weak OOB channel. The SAS messages here are truncated to only 20 bits of the hash of negotiated DH key for better performance. In the end, both sides can verify whether the received SAS and the transmitted SAS matches. If the verification succeeds, the two parties perform key confirmation as depicted in Steps 9 – 12.

4.3 Discussion of MVSec protocols.

In MVSec-I, the vehicle and the smartphone share a short secret key, K_s after Step 2. This key, however, cannot be directly used as a session key for data encryption because of the following reasons. First, K_s is a short key (20 bits). An attacker may perform bruteforce attacks to derive this key. We are sending short key because the low data rate of the OOB channel renders transmission of a longer key (e.g., 128-bit AES key) impractical (discussed in Section 6). Second, the phone does not contribute to deriving the long term secret key. This may be a problem if there is a security bug in the party that is generating the key (e.g., low entropy PRNG for seeding value).

The light in a glove compartment is a unidirectional OOB channel. In order to perform mutual authentication, we provided secrecy in addition to authenticity in the channel, hence a strong OOB channel. However, the sound channel is bidirectional, and therefore, does not require the additional secrecy property.

The aforementioned MVSec protocols achieve the goals mentioned in Section 2. By securing the key agreement messages using different cryptographic schemes along with the use of the OOB channel, the four protocols demonstratively identifies the communicating devices to the human user, successfully establishes the secure channel, and defends against possible MitM attacks. In addition to providing data secrecy, it minimizes the tasks that the user needs to perform, providing a high usability.

5. SECURITY ANALYSIS

We present our analysis on the aforementioned protocols to demonstrate that the protocols successfully defend against MitM attacks. To succeed in a MitM attack, an attacker attempts to inject forged messages to pass the verification steps set forth by the communicating entities. Hence, the attacker needs to break the MAC verification step in MVSec-I, and the hash verification steps in MVSec-II and III.

As a basic attack, the attacker forges the DH public keys to succeed in MitM attacks. (Recall that we do not show *mod p* explicitly for all Diffie-Hellman constructions.) In MVSec-I, the attacker may attempt to forge the encrypted DH public keys (e.g., $\{g^{\tilde{a}}\}_{K_s}$ or $\{g^{\tilde{b}}\}_{K_s}$). However, (s)he fails to suc-

ceed in the attack because both g^a and g^b are protected by the encryption, and the shared secret K between two parties will later be verified by the key confirmation steps by recomputing MACs (e.g., check $M_{K'}(n_A) \stackrel{?}{=} M_K(H(\{g^a\}_{K_s}))$. The attacker also fails to succeed in the MitM attack for MVSec-II and III. To impersonate party A, the attacker sends to B $g^{\tilde{a}}||\tilde{n_A}$ (Step 4 in Figure 3) or $g^{\tilde{a}}$ (Step 4 in Figure 4). The probability that the attacker succeeds is equivalent to finding a hash collision or forging a valid MAC without knowing the key. This probability is bounded by 2^{-l} where l is the bit length of hash or MAC. According to our design, the actual probability is only 2^{-256} since we use SHA-3 as our hash function.

To guarantee that the proposed protocols are secure against the MitM attacks, we perform protocol verification using AVISPA, an automatic validation tool for security protocols. We provide the results at the end of this section.

A more sophisticated attack is to compromise the low entropy 20-bit key for encryption in MVSec-I. An attacker may also compromise the low entropy 20-bit SAS messages in MVSec-II and III. MVSec-I uses the low-entropy K_s to protect the DH public keys by symmetric encryption. An attacker, however, may attempt to impersonate A by guessing K_s in the following manner. The attacker first encrypts its DH public key using the guessed key, and transmits $\{g^{\tilde{a}}\}_{\tilde{K_s}}$ to B. B then decrypts the message using the legitimate K_s and obtains an random value, $g^{a''}$. B now computes the DH session key, $K'' = (g^{a''})^b$ and encrypts B's DH public key using the legitimate K_s and transmits $\{g^b\}_{K_s}$ along with the key confirmation MAC $M_{K''}(n'_A)$. The attacker receives these values, but has no way to compromise K_s because the MAC is generated by the negotiated key K'', which is unpredictable for the attacker. Therefore, the attacker can only guess K_s blindly; the success probability of correct guess is only 2^{-20}.

In MVSec-II, the low-entropy number is sent through the weak OOB channel. If the attacker attempts to launch a MitM attack on both entities, (s)he needs to forge two nonces $\tilde{n_A}$ and $\tilde{n_B}$ with forged negotiated DH keys, $\tilde{K}$ and $\tilde{K'}$ (See Figure 3). However, the attacker's success probability is still only 2^{-20} because the legitimate entities exchange the SAS messages $n_A \oplus n_B$ and $n_B \oplus n_A$ to ensure that the received n_A and n_B are correct. This confirmation also validates the authenticity of the received DH public keys g^a and g^b. We make a similar analysis for MVSec-III. The attacker has just 2^{-20} success probability to forge bogus g^a because $[H(K)]_l$ and $[H(K')]_l$ (where l is 20 bits) are exchanged via a weak OOB channel.

Protocol Analysis by AVISPA We verify all proposed protocols through AVISPA [32], a state-of-art automated validation tool. We first translate all MVSec protocols into the HLPSL language format. In each HLPSL file, we define two entities, *Alice* for the vehicle, and *Bob* for the phone. Actions of the protocols are defined via state machine transitions. All channels in AVISPA are Dolev-Yao channels (under an assumption that an attacker could eavesdrop, intercept, and drop any messages). For the strong OOB channel, we simulate the channel by leveraging a secret pre-shared by *Alice* and *Bob*, which is concealed from the attacker. For the weak out-of-band channel, we simulate it by providing only authenticity property via MACs generated with a pre-shared secret that the two parties share but remain unrevealed to the attacker.

At the end of the HLPSL file, we define two security goals, authenticity and secrecy, and validate whether the proposed protocols achieve these goals. To describe how we translate the abstract protocol to HLPSL format, we provide HLPSL files on sourceforge [1]. For the analysis, we perform AVISPA verifications over two modes, OFMC (On-the-Fly Model-Checker) and CL-AtSe (CL-based Attacks Searcher).

The OFMC checker represents the given protocol based on symbolic techniques. The checker finds protocol falsifications and ensures correctness. Ideally, OFMC builds a tree structure of the given protocol without bounding the messages an attacker can generate. During the verification, OFMC visits every tree node to ensure the absence of errors or existing attacks based on the attacker knowledge. Another analyzer, CL-AtSe, checks to see if there are possibilities of attacks for a loop-free protocol. CL-AtSe builds constraint rules (CL) according to the given attacker knowledge and searches for all constraints of the protocol steps and verifies if it is possible for an attacker to break the aforementioned security goals.

The verification results are presented in Table 2, where all proposed protocols are validated, satisfying the security goals. We also demonstrate the individual results for OFMC and CL-AtSe analyzers. Under OFMC, MVSec-II and III result in taller trees with more than 10000 nodes. For CL-AtSe, we also verify all proposed protocols. MVSec-I, II, and III complete verification on all analyzed states. In conclusion, all proposed protocols are validated and are shown to be secure by AVISPA.

6. IMPLEMENTATION

We demonstrate working MVSec protocols using the Android platform. We use two Motorola Droid 1 phones running Android 2.2.3 (Froyo) - one to simulate the car and the other to represent the driver's smartphone respectively.

6.1 MVSec Pairing Walk-Through

We now provide a walk-through of the MVSec pairing protocols by describing our implementation prototypes leveraging both the weak and strong OOB channels (described in Section 4.2).

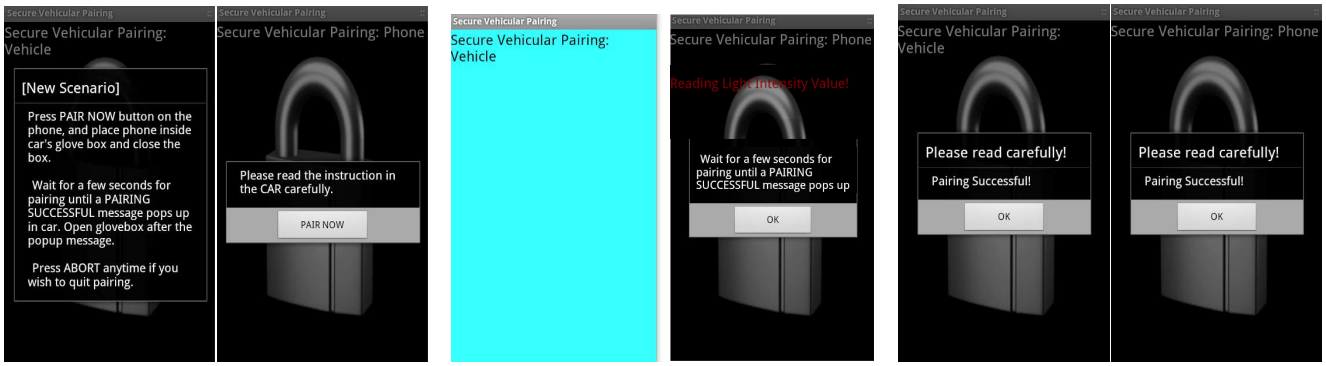
Figure 5 presents chronological protocol steps for the protocols that leverage sound which is a weak OOB channel. (described in Section 4.2). First, both devices prompt the user with instructions and a "Pair Now" button on the car. Once the user presses this button, the two devices will first initiate pairing messages over Bluetooth, as shown in MVSec-II and III protocols. Then the two devices transmit each other's SAS messages over the authentic OOB channel, which is via the audio channel. As soon as the car finishes emitting the beep, the smartphone starts beeping, and the car listens for the beep. In Section 6.2, we present a detailed description of the encoding and decoding of the sound pulses. After the SAS messages have been exchanged, the two devices complete the protocol by exchanging the key confirmation messages again over the in-band Bluetooth channel. After the entire protocol has successfully completed, the two devices prompt the user of a pairing successful message.

Figure 6 provides a similar chronological protocol steps, but leverages light in a glove compartment which is a strong

[1]The HLPSL files can be found in the following URL. `https://sourceforge.net/projects/mvsecproject/`

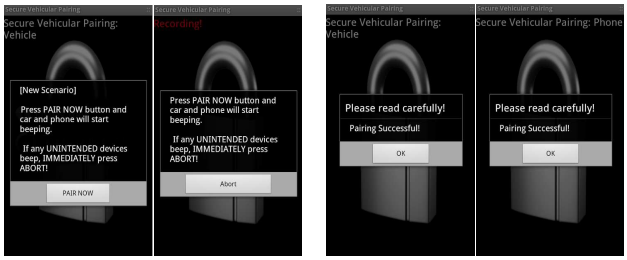
Table 2: AVISPA results for verifying MVSec protocols. All protocols satisfy the security goals set. We present corresponding results such as the number of tree nodes and tree depth in OFMC, and analyzed/reachable states described in CL-AtSe. We also present the execution time for both modes.

Checker	OFMC				CL-AtSe			
Protocols	Authenticity Goal	Secrecy Goal	# of visited nodes (Depth)	Time (Sec.)	Authenticity Goal	Secrecy Goal	# of reachable states	Time (Sec.)
MVSec-I	SAFE	SAFE	120 (6)	0.128	SAFE	SAFE	15/15	0.046
MVSec-II	SAFE	SAFE	51758 (10)	67.803	SAFE	SAFE	1840/1840	0.902
MVSec-III	SAFE	SAFE	32699 (10)	36.179	SAFE	SAFE	7364/7364	13.859



(a) (b) (c)

Figure 6: Execution flow of MVSec protocol leveraging a strong OOB channel (via light in a glove compartment). (a) Upon pressing a start button, the phone simulating the vehicle (car for short) prompts the user instruction to place the phone in the glove compartment. (b) The car is emitting light signal inside the glove compartment and the phone is detecting the signal. (c) User is prompted with pairing successful message.



(a) (b)

Figure 5: Execution flow of MVSec protocol leveraging a weak OOB channel (via sound). (a) Upon pressing a start button, the phone simulating the vehicle (car for short) displays instructions to initiate pairing and to abort if any unintended devices beep. (b) User is prompted with pairing successful message.

OOB channel. Once the user initiates the pairing process, the device simulating the vehicle (car for short) will start transmitting light pulse by varying the light intensity levels of the screen. Section 6.3 also provides implementation detail of the light channel. The smartphone will capture the varying light intensity levels in this step. Then the car and the phone will exchange messages over Bluetooth, to complete MVSec-I protocol. Once the protocol completes, both devices prompt the user of a successful pairing message.

6.2 Audio Channel

MVSec leverages sound as an weak OOB channel for the following reasons. First, as described in Section 4.2.2, the SAS messages only need to be authenticated, but not require to be secret. The audio channel provides authenticity because the driver can easily determine the source device of the sound beeps inside the car - i.e., whether the beep is originating from the car speakers and his intended mobile device, as opposed to other unintended devices (e.g., passenger smartphone). Second, the necessary hardware are already available in the car and the phone, which satisfies the constraints mentioned in Section 2.1. This is because all cars and phones have speakers and microphones.

In order to transmit 20 bits of the SAS message, we first encode the data into eight different frequencies, allowing 20 bits to be encoded to 8 pulses. It takes roughly 800 ms to transmit a pulse (including the pause), so it takes roughly 5.6 seconds to transmit all 20 bits of data. For example, when the transmitter transmits 0x93759 as the SAS message, it is first encoded the message to '2233531' in base 8. On the receiver's side, we leverage Android's AudioRecord class to record the sound signal. Once the signal is recorded, we filter the signal by applying Goertzel algorithm [33, 34] for the eight target frequencies. We use eight frequencies evenly distributed from 900Hz to 1600Hz. The aggregate of the filtered frequencies represented by their magnitude squared (mag^2), is depicted in Figure 7. Each spike represents the pulse that correlates to a base 8 number. To finalize the decoding phase, we process the pulses by applying a sliding window technique to the mag^2 values. The sliding window algorithm is triggered when the mag^2 value exceeds a certain threshold, th. Upon triggering the sliding

window algorithm, we check to see if the mag^2 value exceeds th within a certain window size, wnd. If the value exceeds th, we increment a counter until it exceeds the detection threshold, dth. We then classify this window as a legitimate sound pulse. Using empirical analysis, we set wnd=4000, th = 40, and dth=200. The described processing increases the detection accuracy, and reduce false positives, and successfully decodes the pulses to the correct '2233531'(0x93759).

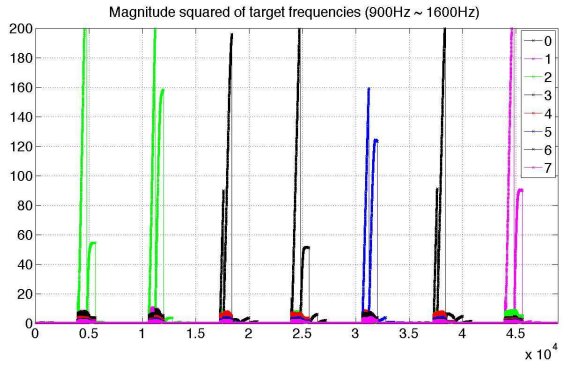


Figure 7: Magnitude squared of target frequencies 900Hz - 1600Hz. The decoding algorithm will process this to '2233531' (== 0x93759).

6.3 Visual Channel

MVSec leverages a strong OOB channel to transmit a short, temporary secret key to defend against the MitM attack. This OOB channel leverages the light bulb in a closed glove box to transmit messages, which will be detected by an ambient light sensor on the driver's smartphone. An Android smartphone is equipped with an ambient light sensor that measures the light intensity experienced by the phone. This sensor is generally used to detect light intensity for automatic brightness control and screen locking. We leverage Android's SensorManager class to implement the prototype using the ambient light sensor. In our implementation, we fully implement the driver's smartphone, and simulate the car's glove box light source, by using another Android device, by varying the light intensity of the screen.

When the driver presses the start button on each device to initiate the protocol, the car will emit a sequence of light signals to the driver's smartphone. The signal is an encoding of a short temporary key (20 bits) as described in the protocol details in Section 4. Accounting for the low resolution of the ambient light sensor on the smartphones, the current prototype leverages four intensity levels to encode the corresponding bits: low, medium, high, and pause. Each level corresponds to the following lux values received by the receiver's ambient light sensor – 10 lx, 40 lx, 90 lx, and 160 lx. Due to the low sampling rate of the ambient sensor in the driver's phone, the car transmits one intensity level for every two seconds (one second for intensity value and another second for the pause bit), and takes a total of 26 seconds to transmit (20 bits encodes to 13 pulses). However, we envision that more responsive ambient sensors installed in newer phones will increase the speed.

MVSec uses the ambient light sensor as a proof-of-concept. However, we envision that using other sensors to read the light signal (e.g., camera) would increase the overall detection time and improve the performance.

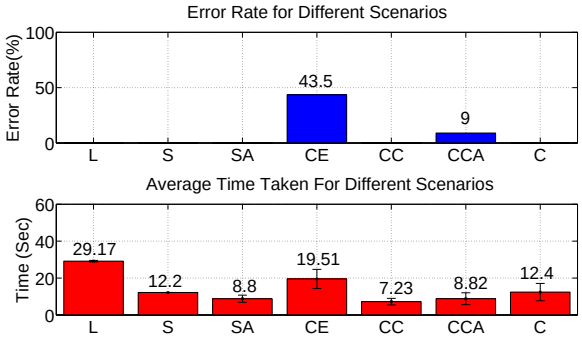


Figure 8: Error rate and time measurements of different study types. Note attack scenarios for both *S* and *CC* are included.

7. EVALUATION

This section provides the evaluation of the usability, as well as the OOB channel detection accuracy. We present the results of the user study conducted by describing the participant profile, study process, and analysis of the results. We also evaluate how accurate the OOB channel is in terms of the detection accuracy.

7.1 Usability Analysis

The main goal of this user study is to determine the usability of the MVSec. Specifically, we design our study to verify (1) whether MVSec reduces user errors as well as pairing timing, and (2) the user's perception of MVSec being more secure and simple to use compared to other solutions.

7.1.1 Demographics

We recruited 23 participants from different sources such as Craigslist and a university mailing lists. The participant pool varied in gender, age, and education background. The participants' age range was 20–59; 13 are in twenties, 6 are in thirties, and 4 are in more than forties. These participants include 12 male and 11 female. Among 23 participants, 10 have undergraduate degree (e.g., master or doctorate degrees), 13 have college degree, and one participant has only high school diploma.

7.1.2 User Study Process

Participants are invited to the driver's seat in a car to perform user study. We present to them with two phones – one to simulate the car's control unit (P_{car}) and the other to be used as the driver's smartphone (P_{driver}). P_{car} is attached to the car's dashboard to simulate the vehicle's infotainment system. We designed both the *light (L)* and *sound (S)* MVSec scenarios to be tested for the user study. Although we fully implemented the working prototype for L scenario as mentioned in Section 6, we simulated the L scenario for the user study by asking the user to place P_{driver} into the glove compartment and explained to them that the light in the compartment will be emitting secret light signal to P_{driver}.

For comparison, we implemented three baseline cases that are currently used as Bluetooth pairing schemes in vehicles.

Table 3: Different scenarios presented to the participants.

Scenarios	Description
Light (L)	Light in a glove box. MVS-I is represented by this scenario. Participant are informed that the light in the glove compartment will transmit secret message to the smartphone's ambient light sensor.
Sound (S)	Bi-directional OOB channel using audio signals. MVS-II and III are represented by this scenario.
Sound Attack (SA)	An unintended device beeps while presenting the *Sound* scenario, and the user is expected to abort after noticing the beep.
Choose-and-Enter (CE)	Base line case that asks the user to create a passkey and enter it on both devices.
Compare-and-Confirm (CC)	Base line case that asks the user to compare the displayed numbers on both devices.
Compare-and-Confirm Attack (CCA)	Different numbers are displayed in the *Compare-and-Confirm*, and the user is expected to notice the difference.
Copy (C)	Base line case that asks the user to copy a displayed number from one device to the other.

The three cases are *choose-and-enter (CE)*, *compare-and-confirm (CC)*, and *copy (C)*. *CE* allows the user to choose a hard-to-guess number and enter it on both of the devices. *CC* allows the user to compare the numbers displayed on each of the devices. *C* allows the user to copy a displayed number on the car, and input it into his phone.

In addition to these five scenarios, we also added two attack scenarios – one for MVSec and the other for the baseline case. First, we present an attack on *sound* by having an unintended device beep, when the participant is performing sound pairing scenario, and test if the participant is able to detect the beep from the unintended device and aborts the pairing process. Second, we present an attack on *CC* by presenting two numbers that are different by a digit, and test if the participant can determine the difference. To reduce bias between the subjects, we present the seven scenarios in random order for different participants. Table 3 lists all seven scenarios presented to the participants.

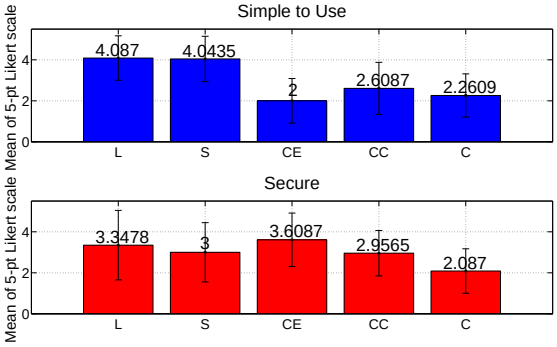


Figure 9: Post-test questionnaire results that rate user's perceptions for simplicity/security.

7.1.3 Study Results

During the execution of the scenarios, we measure the following two outputs for comparison – *error rate* and *time*. For non-attack scenarios (i.e., *L*, *S*, *CE*, *CC*, and *C*), we claim that an error occurs when the participant performs tasks in an incorrect manner resulting in an unsuccessful pairing. For attack scenarios for *S* and *CC*, an error occurs if the participant does not detect a problem, and continues the pairing procedure without aborting. Figure 8 depicts the comparison of the six scenarios with respect to error rate and timing.

The first graph in Figure 8 illustrates that the error rate is around 45% for *CE*, which is a significant percentage. This is because many participants chose easy-to-guess six digit number, when asked to come up with a six digit passkey. Because the security of this approach depends on the passkey to be unpredictable, this demonstrates a clear security problem. We also observe that for the attack scenario of *compare-and-confirm* (Scenario *SA*), about 10% of the participants mistakenly accepted different values displayed on the devices to be the same. However, we did not find any error caused by the participants when pairing via the *L* and *S*. More interestingly, during the attack scenario of *S*, all participants distinguished the beeps from the intended devices as opposed to the unintended device, and pressed abort button as instructed.

The lower graph in Figure 8 depicts the average time taken for different scenarios. On average, *L* took around 29 seconds, which is the longest to complete, due to the low resolution of the ambient light sensor. *CE* followed *L* with around 20 seconds of average completion time. This is because the participants had to come up with a six digit passkey, and enter the number twice, once on each device. *C* and *S* took about the same time of around 12 seconds. The fastest average completion time was *CC*, because this scenario did not require the user to enter any numbers on the devices.

Upon completion of all seven scenarios, we asked the participants to rate the scenarios (excluding the attack scenarios) with a five point Likert scale for *simplicity* and *security* (scale from 1 to 5: 1 being the least simple/secure unto 5 being the simplest/most secure). Figure 9 depicts the average of five point Likert scale. It is interesting to note that both *L* and *S* have significantly higher average (both above average value 4) than the baseline cases for simplicity, despite the fact that *L* took the longest time to complete. It is also interesting to note that the user perception for security are relatively well distributed among different scenarios, fortifying the fact that the participants well represent average users without security expertise.

With the aforementioned results, we claim that MVSec protocols provide a clear usability advantage over the baseline cases, which are used as industry standards in many of the vehicle-to-mobile pairing schemes. We find that MVSec

Table 4: Comparison of sound signal detection accuracy, when varying the background noise level by performing 50 trials per dB.

Average background noise	Success rate	Human perception of background noise
65 dB	90%	Quiet inside car, with some noise from garage ventilation
75 dB	92%	Music volume is loud, but beeps from phones audible
85 dB	86%	Music volume is very loud, beeps from phones are not audible at times
95 dB	2%	Music volume is extremely loud, beeps from phones are not audible at all

simplifies user experience, while significantly reducing error rate.

7.2 OOB Detection Accuracy

We now present the detection accuracy of the audio channel of our prototype implementation discussed in Section 6.2. The detection accuracy of audio signals, even in the presence of background noise, are important because drivers will be faced with similar situations in real-world use cases.

We conducted the test inside a vehicle, and varied the background noise level, by changing the volume while repeating the same song segments. We measured the average background noise by recording the segment via Skypaw's Multi Measures – Decibel smartphone application [35]. We transmitted a 20 bit number via the audio channel between two smartphones separated by 11 inches apart. We repeat the transmission for 50 trials per each background noise level. The test results are summarized in Table 4. As shown in the table, the detection rate is significantly high even when exposed to a relatively loud background noise. However, we find that the detection accuracy drops significantly when the noise level reaches a threshold where the beeps from the phones are almost inaudible to human ears. From this experiment, we claim that the sound detection of MVSec prototypes are robust to be used in practice.

8. DISCUSSION

We now discuss some of the relevant points that were not addressed in the above sections.

Alternative Pairing Methods: There are alternative pairing solutions that may seem to be valid at a first glance for performing a secure key agreement. However, we provide reasons for why they may not be adequate solutions. First, many cars are already equipped with built-in iPod jacks. While it is possible to perform secure key agreement using such cables, we find that not all existing cars today have such cables. We design MVSec to be deployed in all cars, including existing cars without such cables. Second, NFC may be used as an OOB channel to perform authentication. However, NFC suffers the same issue – not all cars are equipped with NFC chips today. Furthermore, many smartphones and tablets including all iOS devices ship without NFC chips. We find that NFC cannot meet our goal of deploying MVSec to all existing cars, while incurring minimal hardware cost.

Visual Channel: Recall that our solution leveraging visual channel was established by varying the light intensity in the glove compartment to emit signals to the phone inside the compartment. While current cars today only have a simple mechanical controller that turns on the light when the compartment door opens, we envision that the light source can be controlled by either installing a new ECU (Electronic Control Unit) or being controlled by existing ECU in the future. To support MVSec in existing cars, dealers can easily service existing cars to install such controllers.

Access Control Policy: MVSec employs an access control policy where the right to drive the vehicle equates to the right to pair a phone. In addition, the driver may delegate such rights to the passengers. However, there may be situations that such policy may not be sufficient. This is best exemplified when the driver leaves his car with valet parking or repair service center. If the glove compartment is unlocked, the valet or service personnel may pair their phones with the car. To resolve this issue, we envision MVSec to employ the following mechanism. MVSec may enforce the car to prompt the driver's phone for any additional pairing requests, so that the car would only proceed with the pairing process after the driver's authorization. (We assume that first phone to be paired does not require such authorization.)

Bluetooth Discovery Overhead: MVSec aims to increase usability while guaranteeing security. One of the main usability drawbacks of the current Bluetooth pairing process is the slow device and service discovery phases. In addition to using the OOB channel for security, we may leverage the OOB channels to transmit the vehicle's Bluetooth MAC address, eliminating human efforts in Bluetooth pairing setup. Thus, the only task the user has to perform is clicking a button on a vehicle and a smartphone. The task will take less than 13 seconds to transit a 48-bit long Bluetooth mac address using sound signals in the current implementation.

Potential OOB Channels in Future: We find that newer cars will be equipped with various types of sensors and actuators, which can be used to establish alternative OOB channels. For example, the vehicular industry is moving towards installing haptic seats (using vibrators) and accelerometers. These new equipment can help establishing an alternative weak OOB channel with a smartphone, as the phone can receive messages encoded via the vibration signals using its accelerometer, and also transmit its signal encoded using vibration signals as well.

9. CONCLUSION

Wireless device pairing is often vulnerable to MitM attacks. Thus, secure pairing between a vehicle and a phone is important for a successful industry deployment. The proposed protocols in this paper address solutions to protect against these attacks, while providing demonstrative identification to the human user. MVSec leverages readily available hardware to allow a car and a phone to perform secure key agreement without any pre-shared secret, and independent of a trusted third party, while still preserving usability.

10. REFERENCES

[1] K. Scarfone and J. Padgette, "Guide to bluetooth security," *NIST Special Publication*, vol. 800, p. 121, 2008.

[2] K. M. Haataja and K. Hypponen, "Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures," in *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*. IEEE, 2008, pp. 1096–1102.

[3] M. Barbeau, J. Hall, and E. Kranakis, "Detecting impersonation attacks in future wireless and mobile networks," in *Secure Mobile Ad-hoc Networks and Sensors*. Springer, 2006, pp. 80–95.

[4] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *NDSS*, 2002.

[5] C. Kuo, J. Walker, and A. Perrig, "Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup," in *USEC*, 2007.

[6] B. C. S. W. Group, "Bluetooth simple pairing Whitepaper," Bluetooth SIG Whitepaper V10r00, Aug. 2006.

[7] Y. Shaked and A. Wool, "Cracking the bluetooth pin," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 39–50.

[8] S. Viehbock, "Brute forcing Wi-Fi Protected Setup. When poor design meets poor implementation." http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf, Dec. 2011.

[9] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *USEC*, 2007.

[10] C. C. Consortium, "Mirror Link," http://www.mirrorlink.com/.

[11] A. Inc., "Apple CarPlay - The Best iPhone Experience on Four Wheels," http://www.apple.com/ios/carplay/.

[12] H. McCracken, "Apple's iOS in the Car Is Arriving, and ItÕs Called CarPlay," http://time.com/11315/apple-carplay/.

[13] F. M. Company, "MyFord Touch Defines Intuitive Driver Experience: Advanced Capabilities All Voice-Controlled Now," corporate.ford.com/news-center/press-releases-detail/pr-myford-touch-defines-intuitive-31716.

[14] "Zipcar - Wheels When You Want Them," http://www.zipcar.com/.

[15] "Mobility - Car Sharing," http://www.mobility.ch.

[16] "Lyft - Your Friend With a Car," http://www.lyft.me/.

[17] M. Lynley, "Meet Lyft, A Startup Trying To Change San Francisco's Decades-Old Transportation System," http://www.businessinsider.com/lyft-ride-sharing-john-zimmer-2012-9.

[18] R. McMillan, "'Car Whisperer' Puts Hackers in the Driver's Seat," http://www.pcworld.com/article/122077/article.html, Aug. 2005.

[19] T. Group, "Car Whisper," http://trifinite.org/trifinite_stuff_carwhisperer.html, 2004.

[20] W. Alliance, "Wi-fi protected access: Strong, standards-based, interoperable security for today's wi-fi networks," *Retrieved March*, vol. 1, p. 2004, 2003.

[21] J. McCune, A. Perrig, and M. Reiter, "Seeing-is-believing: Using camera phones for human-verifiable authentication," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[22] C. Soriente, G. Tsudik, and E. Uzun, "HAPADEP: human-assisted pure audio device pairing," *Information Security*, pp. 385–400, 2008.

[23] C. Kuo, M. Luk, R. Negi, and A. Perrig, "Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes." in *ACM Conference on Embedded Networked Sensor System (SenSys)*, Nov 2007.

[24] C. Gehrmann, C. J. Mitchell, and K. Nyberg, "Manual authentication for wireless devices," in *RSA Cryptobytes*, 2004.

[25] E. Uzun, N. Saxena, and A. Kumar, "Pairing devices for social interactions: a comparative usability evaluation," in *Proceedings of the 2011 annual conference on Human factors in computing systems*. ACM, 2011, pp. 2315–2324.

[26] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "A comparative study of secure device pairing methods," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 734–749, 2009.

[27] R. Kainda, I. Flechais, and A. Roscoe, "Usability and security of out-of-band channels in secure device pairing protocols," in *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 2009, p. 11.

[28] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1992.

[29] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings." in *International Cryptology Conference (CRYPTO)*, 2005.

[30] S. Pasini and S. Vaudenay, "Sas-based authenticated key agreement," in *Theory and Practice of Public-Key Cryptography (PKC)*, 2006.

[31] S. Laur, N. Asokan, and K. Nyberg, "Efficient mutual data authentication using manuallyauthenticated strings." in *Cryptography and Network Security (CANS)*, 2006, pp. 90–107.

[32] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani *et al.*, "The avispa tool for the automated validation of internet security protocols and applications," in *Computer Aided Verification*. Springer, 2005, pp. 135–165.

[33] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *American Mathematical Monthly*, vol. 65, pp. 34 – 35, 1958.

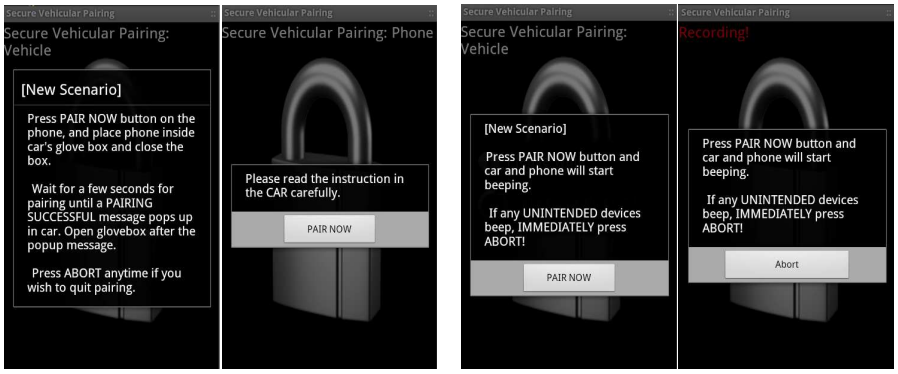
[34] E. Cheng and P. Hudak, "Audio Processing and Sound Synthesis in Haskell," Jan. 2009.

[35] SkyPaw, "SkyPaw Multi Measures – Decibel," http://www.skypaw.com/apps/multimeasures/, 2012.

APPENDIX

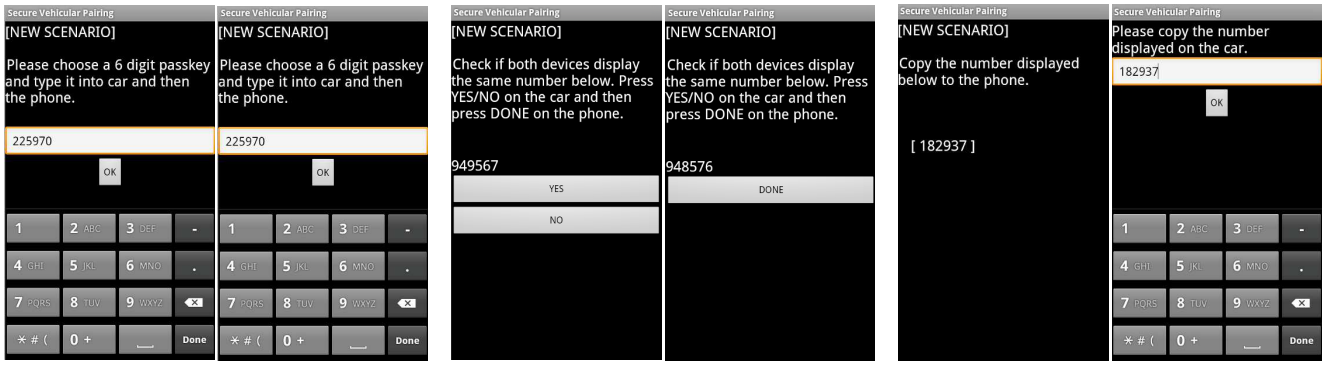
A. USER STUDY INSTRUCTIONS

We present the screenshots of the user study described in Section 7.1. Figures 10 and 11 illustrate the scenarios that are described in Table 3. Figure 12 shows the success and failure messages that are prompted to the user at the end of each scenario.



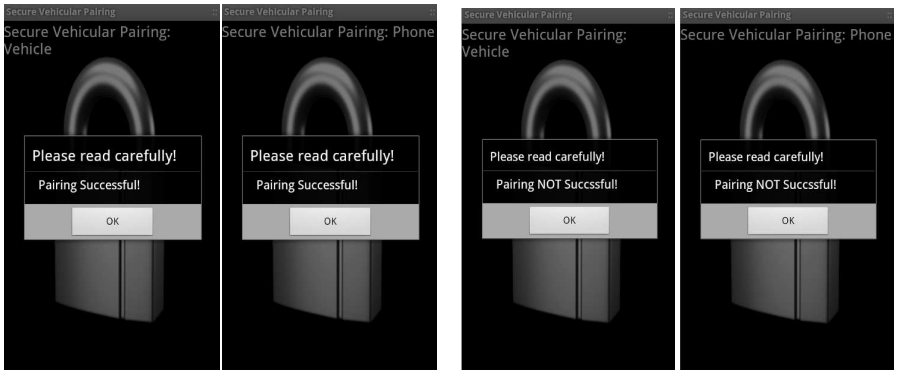
(a) Light (L) using strong OOB (b) Sound (S) using weak OOB

Figure 10: MVSec scenarios - Light and Sound.



(a) Choose-and-Enter (CE) (b) Compare-and-Confirm Attack (CCA) (c) Copy (C)

Figure 11: Three different baseline cases.



(a) Pairing Success (b) Pairing Failure

Figure 12: Pairing successful/failure messages