5-2021

On the root of trust identification problem

Ivan De Oliveira NUNES

Xuhua DING
Singapore Management University, xhding@smu.edu.sg

Gene TSUDIK

On the Root of Trust Identification Problem

Ivan De Oliveira Nunes
UC Irvine

Xuhua Ding
Singapore Management University

Gene Tsudik
UC Irvine

ABSTRACT

Trusted Execution Environments (TEEs) are becoming ubiquitous and are currently used in many security applications: from personal IoT gadgets to banking and databases. Prominent examples of such architectures are Intel SGX, ARM TrustZone, and Trusted Platform Modules (TPMs). A typical TEE relies on a dynamic Root of Trust (RoT) to provide security services such as code/data confidentiality and integrity, isolated secure software execution, remote attestation, and sensor auditing. Despite their usefulness, there is currently no secure means to determine whether a given security service or task is being performed by the particular RoT within a specific physical device. We refer to this as the Root of Trust Identification (RTI) problem and discuss how it inhibits security for applications such as sensing and actuation.

We formalize the RTI problem and argue that security of RTI protocols is especially challenging due to local adversaries, cuckoo adversaries, and the combination thereof. To cope with this problem we propose a simple and effective protocol based on biometrics. Unlike biometric-based user authentication, our approach is not concerned with verifying user identity, and requires neither pre-enrollment nor persistent storage for biometric templates. Instead, it takes advantage of the difficulty of cloning a biometric in real-time to securely identify the RoT of a given physical device, by using the biometric as a challenge. Security of the proposed protocol is analyzed in the combined Local and Cuckoo adversarial model. Also, a prototype implementation is used to demonstrate the protocol's feasibility and practicality. We further propose a Proxy RTI protocol, wherein a previously identified RoT assists a remote verifier in identifying new RoTs.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

ACM Reference Format:
Ivan De Oliveira Nunes, Xuhua Ding, and Gene Tsudik. 2021. On the Root of Trust Identification Problem. In *Information Processing in Sensor Networks (IPSN' 21), May 18–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3412382.3458274

1 INTRODUCTION

In recent years, there has been a growing demand, from both industrial and research communities, for Trusted Execution Environments (TEEs) to aid security-critical applications. While TEEs vary widely in terms of architecture, implementation, and functionality, they provide (at least in the idealized model) an isolated execution space offering both code and data integrity, without relying on any assumptions about applications or operating systems. We refer to these functionalities as TEE services. Security of TEE services (among other trusted services) rely on dynamic Roots of Trust (RoTs) to prove their integrity. RoTs consist of minimal trusted components (e.g., trusted hardware as in TPM and Intel SGX, or trusted software as in hypervisors) used to bootstrap and dynamically verify trust in the system as a whole.

Despite the popularity of such services, it is somewhat surprising that there are no "off-the-shelf" means to securely bind a given RoT to the specific physical device housing this RoT. In particular, it is easy to verify that a service is indeed performed by ***some*** RoT. However, it remains a challenge to determine if the service is performed by *the* RoT residing inside a specific physical device. We refer to this problem as Root of Trust Identification (RTI).

To further illustrate and motivate RTI, consider the following sensor auditing scenario highlighted in [1]. A device (e.g., a smartphone) keeps a TEE-enabled secure log of its audio and video (camera and microphone) activity in order to allow after-the-fact auditing. For example, the host of a confidential meeting uses her trusted verifier device to verify that microphones and cameras of attendees' smartphones remain turned off. The technique proposed in [1] consists of using each attendee device's TEE to assure (e.g., via remote attestation) the verifier of the integrity of sensor usage logs on that device. We argue that – even with TEE-based integrity assurance – the attendee can still use his device's microphone/camera and fool the verifier by supplying logs from a remote accomplice device (also equipped with a TEE of same type) that indeed turns off the sensor during the meeting. The response appears to be valid and there is no means for the host to differentiate between replies from the accomplice device and the one presently held by the malicious attendee. Using a dedicated physical channel (e.g., a cable) between the verifier and the attendee's device does not solve the problem as the device may use another channel to communicate with its accomplice.

Another scenario relevant to RTI occurs whenever some malware has been found on a device. A natural course of action is to force one or more of: (i) re-set, (ii) update software, or (iii) erase the device. However, none of these is trivial since the same adversarial behavior can fool the user into believing that her device has been re-set/updated/erased, while in fact it is some other device that has performed those actions.

Due to lack of RTI solutions, attacks of this type are applicable to any TEE-dependent application which assumes that the TEE indeed resides on the device of interest. More genrally, it applies to any service relying on physical presence of an RoT (either hardware-based or software-based) within a particular device. A successful RTI verification can bind the public-key used by the RoT for remote attestation with its hosting device. However, the binding only has a

IPSN' 21, May 18–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8098-0/21/05.
https://doi.org/10.1145/3412382.3458274

Table 1: Notation summary

Notation	Description
Dev-A, Dev-B, Dev-C, ...	Physical devices (e.g., smart-phone, laptop) A, B, C, ...
$\mathcal{R}oT_A$, $\mathcal{R}oT_B$, $\mathcal{R}oT_C$, ...	$\mathcal{R}oT$ residing on physical devices A, B, C, ...
$pk_i, sk_i \leftarrow Gen(\mathcal{R}oT_A)$	$\mathcal{R}oT_A$ issues i-th session public-key pk_i, and corresponding secret key sk_i. Anyone can verify that pk_i was generated by some $\mathcal{R}oT$. However, $\mathcal{R}oT_A$ signs pk_i using its master secret key in a group signature scheme, thus one cannot tell whether pk_i was issued by $\mathcal{R}oT_A$ or not.
$Pr[A\|B]$	Probability of event A given that event B is true
$Pr[A\|\neg B]$	Probability of event A given that event B is not true
l	Security parameter
$negl(.)$	a negligible function: $negl(l) \leq 1/2^l$
$\mathtt{BT} \leftarrow \mathtt{BT.Sample}(U, \text{Dev-A})$	Sampling of biometric template BT from user U performed by biometric sensor on physical device Dev-A.
$\mathtt{HD} \leftarrow \mathtt{FV}_{\mathtt{GEN}}(\mathtt{BT}, \mathcal{C}\text{hal})$	Generation of helper data HD from biometric template BT and randomness $\mathcal{C}$hal.
$\mathcal{C}\text{hal}' \leftarrow \mathtt{FV}_{\mathtt{OPEN}}(\mathtt{BT}', \mathtt{HD})$	Reconstruction of randomness $\mathcal{C}$hal$'$ from helper data HD and biometric BT'.
$\sigma \leftarrow \text{sign}_{sk}(M)$	Signature result σ of using sk to sign message M. Implicitly we assume sign_{sk} to be a confidentiality preserving signature scheme, i.e., M cannot be extracted from σ
$\text{verify}_{pk}(\sigma) \equiv M$	Verification of signature σ on message M for public key pk.

long-lasting effect for RoTs using a device-specific persistent public key. For those privacy-friendly TEEs that use short-lived public keys certified with a group signature (such as Intel SGX), the binding is ephemeral. Hence, it is imperative to conduct RTI verification on a per-session basis for TEEs with privacy and unlinkability protection.

We observe that many TEE-enabled devices (e.g., laptops, tablets and smartphones) are equipped with biometric sensors connected to the TEE via secure physical channels. Because biometric templates are sensitive and hard to revoke, this secure channel is used to secure the biometric template in case of a compromised application and/or operating system, while still allowing biometric authentication as shown in FIDO [2]. In this paper, we propose a low-burden user-aided approach to RTI. The basic idea is that the TEE vouches for the biometric template securely obtained from the hard-wired sensor. We **do not use biometrics** to authenticate the user. Instead, a biometric is used as a challenge. Security of our approach is based on the difficulty of cloning a human biometric (e.g., a fingerprint) in real-time during RTI verification. However, prior enrollment of a user's biometric is not required. We also do not use the same biometric in different sessions. Because it is used as a challenge, the only properties the biometric needs are: sufficient entropy and (real-time) unclonability, which biometrics used for user authentication are assumed to have.

In the rest of this paper, after formalizing RTI and describing the attack models, we construct a biometric-based RTI scheme. We also prototype and evaluate our scheme using an $\mathcal{R}oT$ based on a trusted micro-hypervisor to demonstrate its practicality. We consider RTI as a subtle and important issue, which has been mostly overlooked.

2 RTI PROTOCOLS

In this section we define RTI protocols and the adversarial model. Our notations are summarized in Table 1. As noted in Section 1, some types of TEEs use a device-specific persistent public key while others use one-time public key with group signature based certification. Without loss of generality, our treatment in this section focuses on the latter type since it subsumes the former.

2.1 Definitions

Suppose Dev-A is a physical device (e.g., smartphone, personal computer, server) equipped with an RoT denoted by $\mathcal{R}oT_A$. Let:

$$pk_i, sk_i, \sigma_i \leftarrow Gen(\mathcal{R}oT_A) \tag{3}$$

denote the process whereby $\mathcal{R}oT_A$ generates the i-th asymmetric key pair (pk_i, sk_i) and a group signature σ_i upon pk_i. Although σ_i can be verified cryptographically, it does not prove that pk_i is for Dev-A, because the signature does not enclose any physically identifiable property of Dev-A.

An RTI protocol is the interactions between a verifier ($\mathcal{V}rf$) and a prover RoT ($\mathcal{R}oT_P$) which issues pk_i and is alleged to reside on Dev-A. Both parties are trusted and cooperate such that $\mathcal{V}rf$ can decide if $\mathcal{R}oT_P$ resides in Dev-A, i.e., whether $\mathcal{R}oT_P \equiv \mathcal{R}oT_A$. Interestingly, not even $\mathcal{R}oT_P$ itself knows its own residency. This goal is deceptively simple and, as we discuss in the remainder of this paper, is hard to achieve even though both parties involved in the protocol are trusted.

At the end of the RTI protocol, $\mathcal{V}rf$ learns pk_i which is a public key used by $\mathcal{R}oT_P$. $\mathcal{V}rf$'s assertion on $\mathcal{R}oT_P \equiv \mathcal{R}oT_A$ also implies that pk_i is indeed issued by $\mathcal{R}oT_A$. Completeness and security of RTI are defined in terms of $\mathcal{V}rf$'s ability to make a positive conclusion if and only if $pk_i \leftarrow Gen(\mathcal{R}oT_A)$, with overwhelming probability. We specify a generic RTI protocol in Definition 1. Completeness

Definition 1 (RTI Protocol).
RTI(A, pk_i) *is a 2-party interactive protocol executed between* $\mathcal{V}$rf *and* $\mathcal{R}$oT$_P$.
$\mathcal{V}$rf *selects a physical device Dev-A and* $\mathcal{R}$oT$_P$ *issues* pk_i *– a session public-key.*
The protocol outputs 1 if $\mathcal{V}$rf *concludes that* pk_i *was issued by* $\mathcal{R}$oT$_A$*; or 0 otherwise.*

Definition 2 (RTI Completeness).
RTI *is complete iff:*

$$Pr[\text{RTI}(\textit{Dev-A}, pk_i)|(pk_i \leftarrow Gen(\mathcal{R}\text{oT}_A))] = 1 - negl(l) \tag{1}$$

where l *is the security parameter and* $negl$ *is a negligible function.*

Definition 3 (RTI Security).
RTI *is secure iff:*

$$Pr[\text{RTI}(\textit{Dev-A}, pk_i)|\neg(pk_i \leftarrow Gen(\mathcal{R}\text{oT}_A))] = negl(l) \tag{2}$$

where l *is the security parameter and* $negl$ *is a negligible function.*

and security of RTI protocols are stated in Definitions 2 and 3, respectively.

Definition 2 states that a complete RTI protocol against $\mathcal{R}$oT$_A$, always outputs '1' if the public-key pk_i given as input to the protocol is indeed generated by $\mathcal{R}$oT$_A$. Definition 3 states that a secure RTI protocol against $\mathcal{R}$oT$_A$, always output '0', if pk_i given as input to the protocol is not issued by $\mathcal{R}$oT$_A$. Note that by Definition 1, $\mathcal{R}$oT$_P$ is defined as the $\mathcal{R}$oT that issues pk_i, thus the following equivalence:

$$[pk_i \leftarrow Gen(\mathcal{R}\text{oT}_A)] \leftrightarrow [\mathcal{R}\text{oT}_P \equiv \mathcal{R}\text{oT}_A]. \tag{4}$$

We now present several possible attacks on RTI protocols to illustrate some subtleties in addressing the RTI problem.

2.2 Attack Vectors

In this section, we discuss several attack scenarios and argue that addressing RTI is challenging. We start by describing a naïve approach to solving RTI and show how it can be attacked trivially. We then gradually increase adversarial capabilities.

2.2.1 Naïve RTI *Protocol.* As shown in [1], a natural way to solve RTI is to challenge whether $\mathcal{R}$oT$_P$ knows sk_i, assuming that $\mathcal{V}$rf has the prior knowledge of $\mathcal{R}$oT$_A$'s ownership of sk_i. The protocol supposes the scenario in Figure 1(a) and proceeds as follows (communication is assumed to take place over a wireless medium):

(1) $\mathcal{V}$rf requests $\mathcal{R}$oT$_P$ public key;
(2) $\mathcal{V}$rf receives pk_i and checks that it was issued by some legitimate $\mathcal{R}$oT by verifying the group signature on pk_i;
(3) $\mathcal{V}$rf issues a random challenge c, encrypts c under pk_i, and sends it to $\mathcal{R}$oT$_P$;
(4) $\mathcal{R}$oT$_P$ issues signs c using its private key;
(5) $\mathcal{V}$rf verifies the signature from $\mathcal{R}$oT$_P$ using pk_i. If valid, it concludes that $\mathcal{R}$oT$_P$ is $\mathcal{R}$oT$_A$ and pk_i is indeed issued by $\mathcal{R}$oT$_A$;

The problem is that the assumption in the naïve protocol barely holds in reality because it is infeasible for $\mathcal{V}$rf to have the prior knowledge of ownership of the key. Hence, $\mathcal{V}$rf cannot distinguish between an interaction with Dev-A and some other $\mathcal{D}$ev* of the same class and equipped with the same $\mathcal{R}$oT type. In particular, an evil-twin adversary $\mathcal{A}$dv can easily convince $\mathcal{V}$rf that pk_i was issued by $\mathcal{R}$oT$_A$ while in fact pk_i is issued by $\mathcal{R}$oT*. As illustrated in Figure 1(b), $\mathcal{A}$dv performs as follows:

(1) $\mathcal{A}$dv intercepts $\mathcal{V}$rf request and forwards it to $\mathcal{D}$ev*;
(2) $\mathcal{A}$dv replies to $\mathcal{V}$rf with $pk_i \leftarrow Gen(\mathcal{R}\text{oT}^*)$, issued by $\mathcal{R}$oT*;
(3) $\mathcal{V}$rf believes that pk_i was generated by $\mathcal{R}$oT$_A$ and completes the rest of the protocol with $\mathcal{R}$oT*;
(4) $\mathcal{V}$rf incorrectly concludes that pk_i was issued by $\mathcal{R}$oT$_A$.

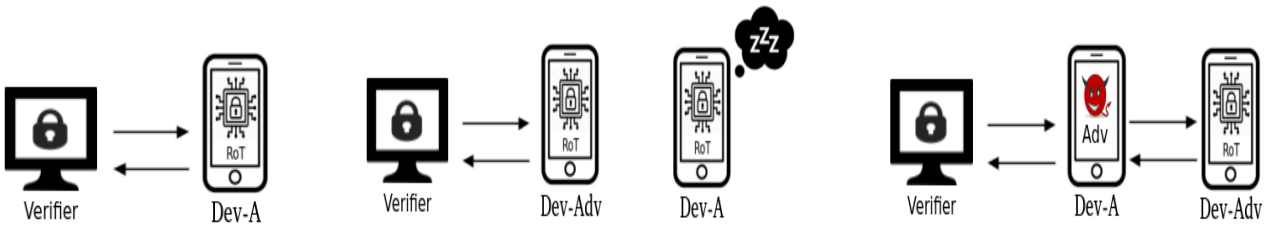
***Remark**: Although $\mathcal{R}$oT* is honest (i.e., not subverted by $\mathcal{A}$dv), it cannot tell that it is being (ab)used by $\mathcal{A}$dv to fool $\mathcal{V}$rf. From $\mathcal{R}$oT* perspective, this interaction is indistinguishable from a legitimate execution of an* RTI *protocol between $\mathcal{V}$rf and itself.*

2.2.2 Coping with Evil-twin Adversaries. One way to cope with an evil-twin adversaries is for $\mathcal{V}$rf to require a physical channel that cannot be tampered with, or accessed, by nearby devices. For example, intercepting $\mathcal{V}$rf messages and replying in place of Dev-A is significantly harder when $\mathcal{V}$rf uses a wired channel (e.g., a USB cable) to communicate with Dev-A. This would prevent $\mathcal{A}$dv from using $\mathcal{D}$ev* to interact with $\mathcal{V}$rf directly, since only Dev-A is directly connected to $\mathcal{V}$rf. In this case, an honest execution of the RTI protocol would proceed as above, except for the use of the wired channel. However, even a wired channel is insufficient if we consider a *cuckoo* adversary [3]. Such an adversary first installs malware on Dev-A. This malware intercepts incoming messages destined for $\mathcal{R}$oT$_A$ and forwards them to $\mathcal{D}$ev*. As illustrated by Figure 1(c), the attack proceeds as follows:

(1) Malware on Dev-A forwards $\mathcal{V}$rf request (received on the direct channel) to $\mathcal{D}$ev*, which feeds to $\mathcal{R}$oT*;
(2) $\mathcal{R}$oT* replies to $\mathcal{V}$rf request. It issues a pk_i and plays its part in the challenge-response protocol with $\mathcal{V}$rf (inadvertently assuming the role of of $\mathcal{R}$oT$_A$);
(3) Response message from $\mathcal{R}$oT* is relayed to $\mathcal{V}$rf by malware on Dev-A, via the direct channel.
(4) $\mathcal{V}$rf incorrectly concludes that pk_i is issued by $\mathcal{R}$oT$_A$.

As in the evil-twin attack case, $\mathcal{R}$oT* is an honest $\mathcal{R}$oT. However, it cannot tell that it is used by $\mathcal{A}$dv to fool $\mathcal{V}$rf.

2.2.3 Cuckoo Adversaries. Cuckoo attacks show that defending against evil-twin adversaries is not enough when malware is in full control of Dev-A. Indeed, the threat of malware is the main reason



(a) Expected setting in benign RTI protocol execution.

(b) An evil-twin $\mathcal{A}$dv uses $\mathcal{D}$ev* to hijack the communication and play the role of Dev-A.

(c) A cuckoo $\mathcal{A}$dv uses malware on Dev-A to relay $\mathcal{V}$rf messages to/from accomplice $\mathcal{D}$ev*.

Figure 1: Possible scenarios during RTI protocol execution

for Dev-A to be equipped with an $\mathcal{R}$oT. On the other hand, because network I/O interfaces typically go through untrusted components (i.e., drivers and OS), malware presence makes a secure physical connection between $\mathcal{V}$rf and Dev-A insufficient for mitigating the RTI problem. Capabilities of a cuckoo attacker are not restricted to the wired interface (e.g., USB). Any I/O device that does not communicate directly to the $\mathcal{R}$oT must pass through an untrusted component and can be used for cuckoo attacks.

As a matter of fact, an $\mathcal{R}$oT could even be used to verify the existence of a software direct secure path (e.g., implemented by a hypervisor) between itself and the I/O interface inside a given device. Then, as a part of the RTI protocol, $\mathcal{R}$oT would only reply to a challenge coming on that particular verified interface. In the cuckoo attack, $\mathcal{R}$oT* (which is an honest $\mathcal{R}$oT) would refuse to reply to the challenge relayed by $\mathcal{A}$dv, because it is not received from the expected and verified wired I/O interface, since Dev-A and $\mathcal{D}$ev* are not directly connected.

Unfortunately, even this setting can be circumvented by a more potent cuckoo $\mathcal{A}$dv which uses an *accomplice challenger* that connects to $\mathcal{D}$ev* via a channel expected by the $\mathcal{R}$oT. Malicious software on Dev-A can forward $\mathcal{V}$rf messages to the *accomplice challenger*. The *accomplice challenger* then forwards to $\mathcal{D}$ev* the same messages sent by $\mathcal{V}$rf to Dev-A, over the expected I/O interface. Since the view of $\mathcal{R}$oT* is indistinguishable from that of an honest execution of RTI, it produces a legitimate response that passes the verification.

Although the channel expected by the $\mathcal{R}$oT in our example is a wire/cable, this attack applies to any I/O interface. Assuming that the *accomplice challenger* has I/O capabilities equivalent to those of $\mathcal{V}$rf, a challenge from $\mathcal{V}$rf can be replayed by the *accomplice challenger* using the same type of channel. Thus, we observe that ***whenever the challenge is conveyed using a machine I/O interface, it can be replayed by another machine with the same I/O capabilities***. This motivates our choice for a biometric-based RTI scheme. The key rationale is that, if a human user becomes a part of the I/O operation, this I/O operation cannot be easily replayed since it requires physical participation by the same person.

2.3 RTI Adversarial Model

Considering the attack scenarios of Section 2.2, we define a strong adversary $\mathcal{A}$dv that can compromise the entire software stack of Dev-A, *excluding the software component of* $\mathcal{R}$oT$_A$ *e.g., a trusted hypervisor loaded and verified by the hardware component of* $\mathcal{R}$oT$_A$. As such, $\mathcal{A}$dv can compromise applications and the operating system. It can intercept, eavesdrop, discard or inject messages on the internal path between Dev-A's I/O interfaces and $\mathcal{R}$oT$_A$.

We assume that $\mathcal{A}$dv has the same capabilities (intercept, eavesdrop, discard or inject messages) on the network. $\mathcal{A}$dv can sense physical surroundings of Dev-A and $\mathcal{V}$rf and record, retransmit, and replay any message, signal or action performed by $\mathcal{V}$rf or Dev-A actuators. In particular, $\mathcal{A}$dv can deploy its own sensors and actuators with I/O capabilities equivalent to those of $\mathcal{V}$rf and Dev-A, in the environment surrounding them. This model accounts for *evil-twin adversaries* as per Section 2.2.

$\mathcal{A}$dv can deploy an accomplice device $\mathcal{D}$ev* equipped with $\mathcal{R}$oT*. The entire software state of $\mathcal{D}$ev* is also under $\mathcal{A}$dv control. These devices might be located in a remote environment where $\mathcal{A}$dv deploys its own sensors and actuators with I/O capabilities equivalent to those of $\mathcal{V}$rf and Dev-A. Malware on Dev-A (controlled by $\mathcal{A}$dv) might, for instance, intercept messages sent from $\mathcal{V}$rf to $\mathcal{R}$oT$_A$, relaying them to $\mathcal{D}$ev*. $\mathcal{R}$oT* might inadvertently reply to malware on Dev-A which then forwards replies to $\mathcal{V}$rf on behalf of $\mathcal{R}$oT$_A$. This model accounts for *cuckoo adversaries*, as discussed in Section 2.2.

We consider hardware attacks to be out of scope of this paper. Specifically, $\mathcal{A}$dv cannot make hardware changes on Dev-A, any hardware-based $\mathcal{R}$oT, or the physically built-in circuit linking a trusted I/O device and an $\mathcal{R}$oT. Protection against physical attacks is considered orthogonal and can be supported by tamper-resistant hardware techniques [4].

2.4 Mitigating RTI via Presence Attestation

The RTI problem is quite similar to that of convincing a *human user* that her own device has an *active* $\mathcal{R}$oT. The latter is referred to as *Presence Attestation* (*PA*) in [5] which proposes several concrete schemes. In addition to convincing the human user that she is interacting with the $\mathcal{R}$oT on her device, *PA* schemes can be extended so that $\mathcal{V}$rf learns the $\mathcal{R}$oT's public key. Therefore, they are one way to address RTI. Unsurprisingly, *PA* schemes also cope with evil-twin and cuckoo attacks. We now overview three *PA* schemes from [5] and discuss their security from the RTI perspective.

2.4.1 Location-based PA. The security premise of location-based *PA* scheme is twofold: (i) $\mathcal{R}$oT securely obtains genuine location of its hosting device, as reported by GPS; and (ii) given sufficient

knowledge about Dev-A's location, the user can manually verify location reported by $\mathcal{R}$oT, perhaps aided by visualization on a map. The essence of this approach is to use the geographic location as the challenge to $\mathcal{R}$oT. However, besides well-known attacks on GPS signaling [6, 7], its main shortcoming is that it cannot differentiate $\mathcal{R}\text{oT}_A$ from $\mathcal{R}\text{oT}^*$, which is sufficiently close to Dev-A so that they report the same readings. Moreover, manual verification of a geographic location does not have high enough accuracy.

2.4.2 Scene-based PA. This scheme uses a (photo of a) scene randomly chosen by the human user as the challenge and requires $\mathcal{R}$oT to report the challenge received over a secure camera interface. As in the location-based scheme, the human user verifies correctness of the $\mathcal{R}$oT response. This scheme is vulnerable to the evil-twin attack where the adversary takes the picture of the same scene and asks $\mathcal{R}\text{oT}^*$ to sign it. Its security is therefore dependent on the human user's ability to differentiate among photos taken by two different devices, which is obliviously not reliable. This scheme is also vulnerable to analog cuckoo attacks, whereby $\mathcal{A}$dv re-renders the scene to an accomplice display such that $\mathcal{D}\text{ev}^*$ can take a genuine photo of it. Given today's hardware technology, it is infeasible for a normal user to distinguish between a photo of a physical scene and a re-production thereof. In both location- and the scene-based schemes, the human user decides on correctness of $\mathcal{R}\text{oT}_A$'s response. From the perspective of RTI, it takes an extra step for the user to notify $\mathcal{V}$rf about her conclusion.

2.4.3 Sight-based PA. Sight-based *PA* scheme does not require any human input. Its security is based on the observation that any message reply in the line-of-sight channel incurs measurable time delay, because the attack includes analog operations which are comparatively time-consuming. In this scheme, $\mathcal{V}$rf and Dev-A run the standard challenge-response protocol using the line-of-sight channel whereby a display "sends" messages to a camera. Using cryptographic means, $\mathcal{V}$rf checks integrity of the response. In addition, by measuring the time to complete the session, it verifies whether $\mathcal{R}\text{oT}_A$ is at the other end of the light-of-sight channel. Note that this scheme requires $\mathcal{R}\text{oT}_A$ to securely obtain the challenge from the camera and securely display the response to the display.

Although it offers stronger security than location- and scene-based schemes, sight-based *PA* is dependent on the current frame-per-second (fps) rate of commodity cameras on modern smartphones. Moreover, sight-based *PA* requires the two participating devices to be physically well positioned through multiple rounds in order to form a high-quality light-of-sight channel.

Summary. Zhang et. al. [5] have shed light on challenges related to $\mathcal{R}$oT and cuckoo attacks, and made attempts to tackle them. We believe that RTI is both harder and more general than the *PA* problem, since RTI does not assume that the average human user possesses sufficient knowledge and expertise to discern ambient properties. Our biometric-based approach relies on the unclonability of human biometrics with high entropy, the same assumption propping up security in biometric authentication schemes.

2.5 Mitigating RTI via Distance Bounding

Distance bounding protocols [8–11] allow a verifier to determine whether its communication peer is within a certain distance (e.g., 30 cm). They are fundamentally different from a RTI protocol because establishing an acceptable distance does not always *identify* the device. Using distance to solve RTI assumes that there is only single device in the range, which does not hold when the distance is large.

There are also implementation issues using a distance-bounding protocol for RTI. Parno et. al [3] have remarked that it is not suited to deal with the cuckoo attack against TPM-based attestation given the slow speed of TPM. Although today's $\mathcal{R}$oT has better performance, the time variance of signature generation remains too large for distance-bounding protocols which only tolerate time errors in several nanoseconds. Moreover, distance-bounding protocols would require all devices of Dev-A's class to be equipped with distance bounding hardware (ultra wide-band radios with high-precision clocks needed for accurate timing measurements) securely wired to the $\mathcal{R}$oT [12]. This is currently not available in commodity devices.

Recently, Dhar et. al. [13] propose to use a trusted device (e.g., a smart USB device) as a proxy attached to the proving device so that a remote verifier detect the cuckoo attack during SGX attestation. Besides the hassle of using a trusted device, this approach relies on a strong assumption that the trusted device attached to an untrusted environment remain intact.

3 BUILDING BLOCKS

3.1 Biometric Features & Template Matching

A Biometric Template (BT) is composed of features uniquely identifying an individual. In a biometric application (e.g., user authentication) a reference BT is usually sampled and stored as part of the enrollment procedure. During authentication, the feature extraction procedure is used to collect a real-time sample BT' from the purported user. If the similarity score between BT' and BT exceeds a pre-defined threshold, they are considered as a matching pair. The method to evaluate the similarity score and the choice of the threshold depend on the particular biometric. A BT corresponding to user U is represented by a set:

$$\text{BT}_U = \{p_1, ..., p_M\}, \tag{5}$$

where $p_1, ..., p_M$ are data points (features) representing unique details of U's biometric. For instance, $p_i \in \text{BT}_U$ for a fingerprint represents the location and orientation of the fingerprint's *minutiae*. *Minutiae* are regions in the fingerprint image where fingerprint lines start, end, merge and/or split. In turn, each *minutiae* p_i is represented as:

$$p_i = (x_i, y_i, \theta_i) \tag{6}$$

where x_i and y_i are Cartesian coordinates for the minutiae location in the fingerprint image and θ_i is the angle of minutiae orientation. In this paper, we focus on the fingerprint biometric modality, since fingerprint sensors are commonly found on commodity devices, such as laptops and smartphones. Nevertheless, similar encoding techniques are applicable to other biometric templates, such as iris scans [14].

3.2 Fuzzy Extractors & The Fuzzy Vault Scheme

A Fuzzy Extractor [15] (FE) is a cryptographic primitive commonly used in biometric systems. FE can successfully extract the same randomness from different noisy samples of the same biometric as

long as these samples are within a certain distance threshold. This fuzziness in the matching allows, for instance, to match biometric samples acquired using different sensors. One popular FE instantiation is the Fuzzy Vault scheme (FV) [16] which is designed to work with BTs represented by data point sets in Eq. 5. An FV scheme consists of two algorithms: FV_{GEN} and FV_{OPEN}. Given a biometric template BT_U the first algorithm generates the corresponding helper data HD which hides a secret k. Given another biometric template BT'_U and HD, the second algorithm can successfully recover k from HD provided that BT'_U matches BT_U. The notion of FV is captured in Definition 4. Security of FV relies on the infeasibility of the polynomial reconstruction problem [17]. Definitions 5 and 6 formulate FV's completeness and (information theoretic) security.

Definition 4 (FV). *A Fuzzy Vault is defined as* $\mathsf{FV} = (\mathsf{FV}_{GEN}, \mathsf{FV}_{OPEN}, \Phi)$, *where* Φ *is a set of parameters* $\Phi = (d, GF(2^\tau), \mathtt{MS}, \mathtt{dist}, \mathtt{w})$:
- d *is the polynomial degree;*
- $GF(2^\tau)$ *is a Galois Field of size* 2^τ;
- MS *is a metric space;*
- dist *is distance function defined over* MS;
- w *is distance threshold;*

FV_{GEN} *and* FV_{OPEN} *are algorithms defined as follows:*
- FV_{GEN}:
  - ***Inputs***: k *and* BT_U, *s.t.,* $|k| = (d+1) \times \tau$.
  - ***Output***: HD
- FV_{OPEN}:
  - ***Inputs***: HD *and* BT'_U
  - ***Output***: k', *s.t.,* $|k'| = (d+1) \times \tau$.

Definition 5 (FV-Completeness).
$\mathsf{FV} = (\mathsf{FV}_{GEN}, \mathsf{FV}_{OPEN}, \Phi)$ *is complete with* w*-fuzziness if for every possible* $k \in GF(2^\tau)^{d+1}$ *and every pair* BT_U, BT'_U *with* $\mathtt{dist}(\mathsf{BT}_U, \mathsf{BT}'_U) \leq w$:

$$\mathsf{FV}_{OPEN}(\mathsf{FV}_{GEN}(k, \mathsf{BT}_U), \mathsf{BT}'_U) = k \tag{7}$$

with overwhelming probability.

Definition 6 (FV-Security).
$\mathsf{FV} = (\mathsf{FV}_{GEN}, \mathsf{FV}_{OPEN}, \Phi)$ *is* p*-information theoretically secure if any computationally unbounded adversary with access to* HD *is able to guess either,* BT *or* k, *with success probability of at most* p.

FV_{GEN} can be implemented by selecting a polynomial P of degree d defined over a field $GF(2^\tau)$ and encoding (or splitting) the secret k into the $d+1$ coefficients (a_i) of P. The resulting polynomial is defined as:

$$P_k(x) = \sum_{i=0}^{d} a_i x^i \tag{8}$$

where coefficients $\{a_0, ..., a_d\}$ are generated from k and can be used to reconstruct k. Since P_k is defined over $GF(2^\tau)$, each coefficient can encode τ bits; this implies that size of a key that can be encoded is a function of the field size and the degree of the polynomial given by:

$$||k|| = (d+1) \times \tau \tag{9}$$

After encoding k as a polynomial P_k, each of the M data points (features) in BT_U is evaluated in the polynomial P_k generating a list of points in a two-dimensional space:

$$L_P = \{(p_1, P_k(p_1)), ..., (p_M, P_k(p_M))\} \tag{10}$$

Note that the field must also be large enough to encode a single feature from BT_U as a single field element. The resulting set L_P is formed by only by points in the polynomial P_k. In addition to L_P, a set of chaff points L_S of size $N >> M$ is generated by randomly selecting pairs $(r_x, r_y) \leftarrow\!\!\$\, GF(2^\tau)^2$, resulting in:

$$L_S = \{(r_{x,1}, r_{y,1}), ..., (r_{x,N}, r_{y,N})\} \tag{11}$$

Finally, L_P and L_S are shuffled together using a random permutation $\pi_\$$ and the result is published as the helper data HD:

$$\mathsf{HD} = \pi_\$(L_P + L_S) \tag{12}$$

Note that HD also includes the set of public parameters $\Phi = \{F, d, l_P, H(k)\}$, where F is the field over which $P_k(x)$ is defined and d is its degree, l_P is the size of BT_U, i.e., the number of points in HD that belong to $P_k(x)$, and $H(k)$ is a cryptographic hash of k allowing one to verify if the correct secret is reconstructed using FV_{OPEN}[1].

The key idea behind security of the FV scheme is that with $d+1$ distinct points $(p_i, P_k(p_i))$ (namely points on $P_k(x)$), one can interpolate $P_k(x)$, retrieve its coefficients and thus recover k. However, to find the *right* $d+1$ points out of the $M+N$ points in the HD is very unlikely. With appropriate choice of M, N, and d the success probability can be made negligible with respect to a desired security parameter.

To reconstruct k from HD using a new biometric template BT'_U, the FV_{OPEN} algorithm applies a distance function (which must be defined according to the biometric type) to select M points from HD which have the shortest distance to the points in BT'_U. If, out of the M selected points, no less than $d+1$ points are indeed on the original the polynomial P_k, they can be used to interpolate P_k and recover k. Otherwise, no interpolation with combinations of $d+1$ points out of M correctly yields P_k and therefore cannot recover k. To determine whether the resulting k is correct, the algorithm compares its hash to $H(k)$ which is in the public help data. FV_{OPEN} rejects BT'_U if not equal; or accepts it otherwise.

The distance threshold w can be used to tune the balance between the false acceptance rate (revealing k to the wrong user) and the false rejection rate (refusing to reveal k to the rightful user). FV does not require ordered data points in the templates, and neither requires all data points to be in both sets. Only $d+1$ data points in BT'_U must be close enough to points in BT_U. The polynomial degree d acts as an accuracy parameter allowing calibration of the scheme to reduce false acceptance by increasing the required number of matching data points.

In this work we use FVs as a cryptographic building block to realize biometric-based RTI protocols. As shown later in Section 4, FV is used to cryptographically bind a random challenge chosen by $\mathcal{V}$rf to the biometric input in RTI execution.

[1] Using a hash function simplifies the implementation, but makes FV's security computational in the size of the output of the hash. The scheme can be fully information theoretically secure by using error correcting codes.

3.3 Hardware Architecture for Biometric Sensing with TEEs

An advantage of biometric-based RTI is that in several types of modern devices, such as smart-phones and laptops, biometric sensors exist and are directly connected ("hard-wired") to the $\mathcal{R}$oT exclusive memory itself, as depicted in Figure 2. it is usually the case that a biometric sensor (e.g., a fingerprint sensor) is directly hardwired to TEE exclusive memory. Therefore, the user's biometric input is not visible to untrusted (and potentially malicious) software on that device, including the operating system. This means that an input biometric, cannot be obtained by an $\mathcal{A}$dv-controlled malware or OS on Dev-A, obviating the need for a trusted software path to be verified by the $\mathcal{R}$oT upon receiving the challenge. Nonetheless, our prototype implementation (see Section 5) also considers the case where this hardware channel is not readily available. In such a case, we show how to establish a secure channel between the biometric sensor and $\mathcal{R}$oT with the help of a small trusted hypervisor.

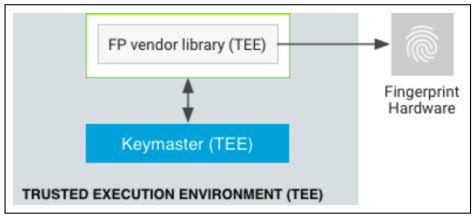


Figure 2: TEE-Biometric hardware architecture of a typical Android device (adapted from [18])

.

4 CONSTRUCTING AN RTI PROTOCOL

We now construct a biometric-based RTI protocol using FVs and analyze its security. We also present a Proxy RTI protocol that can be used to address RTI when $\mathcal{V}$rf is remote.

System Assumption: We assume an authentic (not confidential!) channel between the biometric sensor and the RoT. In some types of devices (e.g., branded smartphones) similar channels are implemented in hardware in order to protect the user's biometric data. Those channels are often claimed by vendors to be both confidential and authentic. Unfortunately, it has been recently shown that biometric data can still be leaked in clever ways[2], which means cuckoo attacks remain possible. In contrast, we believe that it is much harder to compromise authenticity of the channel, since the biometric sensor is hardwired to the RoT. Doing so would imply wholesale RoT compromise. Our scheme is dependent on channel authenticity and unclonability of fingerprints. For devices that do not have this kind of channel, we emulate it in software, by using a micro-hypervisor.

As discussed in Section 2, in a cuckoo attack on the challenge-response RTI protocol, the adversary relays the challenge from $\mathcal{V}$rf. In a conventional challenge-response protocol, a correct response is formed based on two factors: the challenge and the prover's secret. Hence, to counter the challenge relay attack, we include the user in the loop as the third factor needed to produce a correct response. In particular, $\mathcal{V}$rf blinds the cryptographic challenge with the user's biometric by using an FV scheme. $\mathcal{R}\mathsf{oT}_P$ uses its biometric sensor to sample (presumably the same) biometric. The user only provides her biometric to Dev-A's sensor, which can only be read by $\mathcal{R}\mathsf{oT}_A$. Therefore, the only $\mathcal{R}$oT that can unblind the challenge is on Dev-A, which means $\mathcal{R}\mathsf{oT}_P$ is $\mathcal{R}\mathsf{oT}_A$. Since the biometric given to both $\mathcal{V}$rf and Dev-A is the same, if $\mathcal{R}\mathsf{oT}_P$ is not $\mathcal{R}\mathsf{oT}_A$, RTI for Dev-A fails. We now discuss the protocol in more detail.

Remark: *We assume that protocol messages are exchanged over an encrypted and authenticated channel $\mathcal{V}$rf $\leftrightarrow\mathcal{R}\mathsf{oT}_P$. Note that this channel is established between $\mathcal{V}$rf and $\mathcal{R}\mathsf{oT}_P$, i.e., $\mathcal{V}$rf and* **some** *$\mathcal{R}$oT. Even though $\mathcal{R}\mathsf{oT}_P$ has not been identified at this point, it is always possible to check whether pk_i was issued by some $\mathcal{R}$oT. This is necessary to preserve confidentiality of* HD *if a non-reusable* FE *is used to implement the* RTI *protocol. (See Section 6 for further discussion on* FE *reusability.) A secure channel to some (trusted) $\mathcal{R}$oT suffices to preserve confidentiality.*

4.1 FV-based RTI

Figure 3 presents the RTI protocol based on the FV scheme described in Section 3.2. It assumes that $\mathcal{V}$rf and Dev-A are physically accessible to U. U participates in the protocol by providing the same biometric to the sensors of $\mathcal{V}$rf and Dev-A.

The protocol starts with $\mathcal{R}\mathsf{oT}_P$ issuing an asymmetric key-pair and with $\mathcal{V}$rf sampling U's biometric, thus resulting in the template BT_U (line 1). $\mathcal{V}$rf then generates a random l-bit challenge $\mathcal{C}$hal, where l is the security parameter (line 2). Next, $\mathcal{V}$rf uses the FV generation algorithm to obtain HD where BT_U is the biometric and $\mathcal{C}$hal is the secret. $\mathcal{V}$rf sends HD to $\mathcal{R}\mathsf{oT}_P$ (line 3). U also provides the same biometric to Dev-A. As a result, $\mathcal{R}\mathsf{oT}_A$ obtains BT'_U – a new sample of the same biometric (line 4).

Note that the step in line 4 is crucial. Under the assumption of a secure channel between the fingerprint sensor in Dev-A and $\mathcal{R}\mathsf{oT}_A$, BT'_U can only be obtained by the $\mathcal{R}$oT residing in that device, i.e., $\mathcal{R}\mathsf{oT}_A$. If $\mathcal{R}\mathsf{oT}_P$ does not reside in Dev-A, $\mathcal{A}$dv has to provide another biometric to $\mathcal{R}\mathsf{oT}_P$, i.e., from an accomplice person. In such a case, due to FV security, the reconstruction would result in an incorrect $\mathcal{C}\mathsf{hal}' \neq \mathcal{C}\mathsf{hal}$ with overwhelming probability $1 - negl(l)$, for appropriate choice of FV parameters as a function of l. Hence, it would not pass $\mathcal{V}$rf's signature verification (line 7). If verification succeeds, $\mathcal{V}$rf becomes convinced that pk_i is indeed issued by $\mathcal{R}\mathsf{oT}_A$ and $\mathcal{R}\mathsf{oT}_P \equiv \mathcal{R}\mathsf{oT}_A$.

Unlike *PA* schemes, security of our biometric-based RTI scheme is based on $\mathcal{A}$dv's inability to forge BT_U and mount a successful cuckoo attack. Although $\mathcal{A}$dv controls entire software state of Dev-A (except for $\mathcal{R}\mathsf{oT}_A$ itself) and can access any memory outside of that reserved by $\mathcal{R}\mathsf{oT}_A$, it cannot obtain BT'_U due to the secure channel between the fingerprint sensor and $\mathcal{R}\mathsf{oT}_A$.

Fingerprint Forgery: Fingerprints have been used as a biometric for a very long time and remain the most common means of biometric authentication. There have been numerous successful attacks that surreptitiously obtain a user's fingerprints and then come up with various contraptions to fool fingerprint sensors. Clearly, the

[2]See: https://www.blackhat.com/docs/us-15/materials/us-15-Zhang-Fingerprints-On-Mobile-Devices-Abusing-And-Leaking-wp.pdf

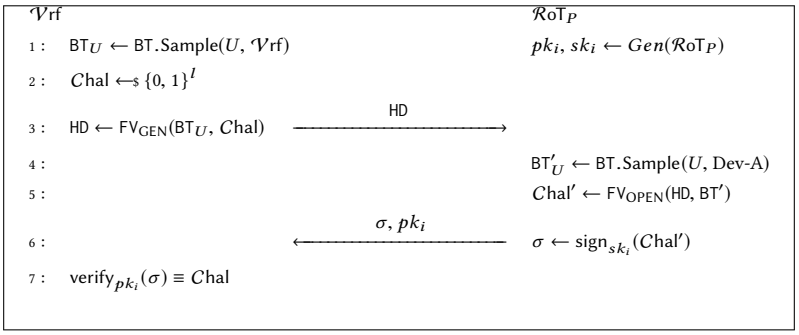


Figure 3: FV-based RTI protocol: $\mathcal{V}$rf decides whether $\mathcal{R}$oT$_P$ resides in Dev-A and, if so, learns its session public-key pk_i.

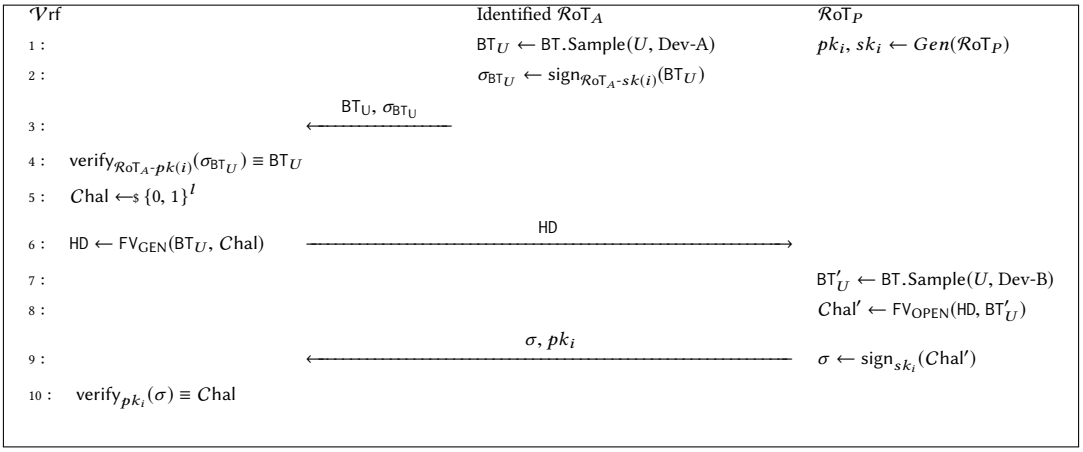


Figure 4: Proxy RTI protocol: $\mathcal{V}$rf is assisted by a previously identified $\mathcal{R}$oT$_A$ (residing on Dev-A) to decide whether $\mathcal{R}$oT$_P$ resides on physical device Dev-B. Dev-A, Dev-B, and user U must be physically co-located. $\mathcal{V}$rf can be remote.

proposed protocol and its variations will fail if the biometric template used in a RTI protocol execution is stolen and reproduced before hand. However, the protocol does not require a pre-determined fingerprint or the user. Hence, the fingerprint forgery attack may not always succeed.

As mentioned above, security of the protocol in Figure 3 depends on that of the FV scheme. Completeness and security of this protocol are stated in Theorems 1 and 2, respectively. In both completeness and security arguments, we assume that whenever two samples are taken from the same biometric they are within a certain distance threshold. Conversely, we assume that two samples of different biometrics are beyond that threshold. In other words, $dist(\text{BT}_U, \text{BT}'_U) \leq w \iff \text{BT}_U$ and BT'_U are samples of the same biometric. In practice, validity of this assumption depends on the accuracy of the biometric matching procedure, including the distance function $dist$, the distance threshold w and the degree of FV polynomial. Our choice of parameters are based on previous work on these issues and are discussed in Section 5. Accuracy results obtained with such parameters are discussed in Section 6.

4.2 Proxy RTI Protocol

The protocol in Section 4.1 requires $\mathcal{V}$rf, and Dev-A to be physically accessible to U, since U must provide her biometric sample to both $\mathcal{V}$rf and Dev-A. To cope with scenarios where $\mathcal{V}$rf (e.g., a server) is not easily approachable, we suggest to use a proxy Dev-A with its $\mathcal{R}$oT$_A$ previously identified, in order to assist $\mathcal{V}$rf in identifying $\mathcal{R}$oT$_B$.

Suppose that U now carries Dev-A to the location of Dev-B. Figure 4 shows a protocol for using Dev-A to assist $\mathcal{V}$rf in remotely identifying $\mathcal{R}$oT$_B$ of Dev-B. The main idea is for $\mathcal{R}$oT$_A$ to act as an interface of $\mathcal{V}$rf. It captures U's biometric and forward it to $\mathcal{V}$rf via an authenticated and secret network channel. The same biometric is also used as a challenge to Dev-B, which runs the rest of the protocol with $\mathcal{V}$rf.

The security of the FV-based RTI protocol in Section 4.1 implies the security of the proxy RTI protocol. We note that Dev-A is *not* a trusted device as used in [13]. Its software, including the OS, could be compromised, while its $\mathcal{R}$oT$_A$ is trusted, which is consistent with the basic protocol. Hence, both protocols provide the same level of security.

As discussed earlier, lack of RTI violates the assumption that $\mathcal{R}$oT resides on the physical device of interest, thus undermining security

Theorem 1. FV-*based* RTI *protocol (Figure 3) is complete according to Definition 2 as long as* FV *is complete according to Definition 5.*

Proof (Sketch) 1. *In an honest execution of the protocol* $\mathcal{R}$oT$_P$ *resides in Dev-A, i.e:* $pk_i \leftarrow Gen(\mathcal{R}oT_A)$.
Since, $\mathcal{R}$oT$_P$ *resides in Dev-A,* $\mathcal{V}$rf *and* $\mathcal{R}$oT$_P$ *(i.e.,* $\mathcal{R}$oT$_A$*) receive* BT$_U$ *and* BT$'_U$ *such that* $dist(\mathsf{BT}_U, \mathsf{BT}'_U) \leq w$. *It follows from Definition 5 that:*

$$\begin{aligned} &\mathsf{HD} \leftarrow \mathsf{FV}_{GEN}(\mathsf{BT}_U, Chal) \rightarrow \\ &Pr[\mathsf{FV}_{OPEN}(\mathsf{HD}, \mathsf{BT}'_U) = Chal] > 1 - negl(l) \rightarrow \\ &Pr[\sigma \equiv sign_{sk_i}(Chal)] > 1 - negl(l) \rightarrow \\ &Pr[verify_{sk_i}(\sigma) \equiv Chal = 1] > 1 - negl(l) \end{aligned} \tag{13}$$

Theorem 2. FV-*based* RTI *protocol (Figure 3) is secure according to Definition 3, as long as* FV *is* p*-information theoretically secure as in Definition 6 and* FV *parameters are chosen such that and* $p = negl(l)$.

Proof (Sketch) 2. *In this case,* $\mathcal{R}$oT$_P$ *does not reside in Dev-A i.e:* $\neg(pk_i \leftarrow Gen(\mathcal{R}oT_A))$.
Therefore, it must be the case that $\mathcal{V}$rf *and* $\mathcal{R}$oT$_P$ *receive* BT$_U$ *and* BT$'_U$ *such that* $dist(\mathsf{BT}_U, \mathsf{BT}'_U) > w$. *Assuming that* $\mathcal{A}$dv *is unable to forge* $sign_{sk_i}(.)$ *with more than* $negl(l)$ *advantage, it follows from Definition 6 that:*

$$\begin{aligned} &\mathsf{HD} \leftarrow \mathsf{FV}_{GEN}(\mathsf{BT}_U, Chal) \rightarrow \\ &Pr[\mathsf{FV}_{OPEN}(\mathsf{HD}, \mathsf{BT}'_U) = Chal] = p = negl(l) \rightarrow \\ &Pr[\sigma \equiv sign_{sk_i}(Chal)] = negl(l) \rightarrow \\ &Pr[verify_{sk_i}(\sigma) \equiv Chal = 1] = negl(l) \end{aligned} \tag{14}$$

of any application dependent on that assumption. The Proxy RTI is itself a good example of such an application. It relies on the assumption that biometric sampling is performed on Dev-A– the device in possession of authorized user U. Therefore, identification of $\mathcal{R}$oT$_A$ is crucial to overall security of this application.

5 PROTOTYPE & EVALUATION

5.1 BT Extraction & FV Parameters

BT extraction generates a biometric template from a fingerprint image. As discussed in Section 3, each data point $p_i \in \mathsf{BT}$ is the position and orientation (x_i, y_i, θ) of a fingerprint minutiae. To extract the BT we use NIST Biometric Image Software (NBIS) [19]. NBIS returns a set of identified minutiae points with corresponding confidence levels. From NBIS output, we select 20 points with the highest confidence and encode them as data points in $GF(2^{24})$. In our prototype, FV's HD is composed of 20 fingerprint data points mixed with 200 random chaff points. The FV polynomial degree is set to 9. Finite field operations are implemented using the Number Theory Library (NTL) [20].

In FV_{OPEN}, the candidate minutiae points are selected from the HD based on their distance to minutiae points in the new template BT$'$ sampled from the user. Similar to [21], we use a distance function between $p_i \in \mathsf{HD}$ and $p'_j \in \mathsf{BT}'$ defined as:

$$D(p_i, p'_j) = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2} + \beta \times \Delta(\theta_i, \theta'_j) \tag{15}$$

where $p_i = (x_i, y_i, \theta)$, $p'_j = (x'_i, y'_i, \theta')$, and $\Delta(\theta_i, \theta'_j) = min(|\theta_i - \theta'_j|, 360 - |\theta_i - \theta'_j|)$. Parameter β controls the degree of importance given to minutiae orientation in computation, as compared to the euclidean distance between the points. A data point p_i is selected if $D(p_i, p'_j) < w$ for some point in $p'_j \in \mathsf{BT}'$. As described in [21], parameters β and w must be empirically calibrated to yield the best accuracy results. Our parameters are empirically calibrated to: $\beta = 0.2$ and $w = 20$. To improve accuracy results for noisy fingerprint readings before extracting the template, during the biometric sampling, we run the fingerprint pre-alignment algorithm from [22]. Figure 5(a) illustrates the result of the template extraction for two pre-aligned fingerprint images. White squares highlight the minutiae points detected in these fingerprints. We discuss the accuracy of this implementation in Section 6.5.

Remark: *We implement our own* BT *extraction to have a fully working prototype and report on its accuracy. We stress that accuracy of the underlying* BT *extraction technique is orthogonal and not affected by the* RTI *setting considered in this work.*

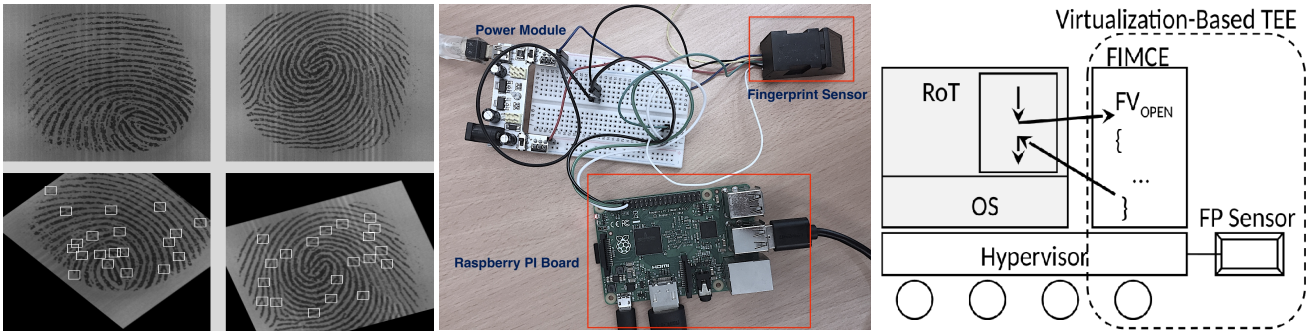
5.2 Prototype

Due to the close environment of hardware-based TEEs with fingerprint sensing (commonly found on mobile phones), we implement the prototype of FV-based RTI on a development board connected with an external fingerprint sensor. The sensor collects user fingerprints and also provides an interface to export the data to a secure storage inaccessible to applications and the operating system. We build a hypervisor-based secure execution environment (software-based $\mathcal{R}$oT) to run $\mathcal{R}$oT$_P$ steps in the FV-based RTI protocol.

5.2.1 Hardware Setting. Figure 5(b) shows the hardware setting of our prototype. An FMP12 Optical Fingerprint sensor is connected to the Raspberry Pi 2 development board with four Cortex-A7 CPU cores at 800 MHz and 1 GB main memory. It runs Debian Linux with kernel version 3.18.8. Software on the board can use a serial port mapped at physical address 0x3F201000 to issue commands to the fingerprint reader and read the collected data.

5.2.2 Virtualization Based $\mathcal{R}$oT. We harness virtualization techniques to build an $\mathcal{R}$oT secure against attacks from the operating system. Our secure environment shown in Figure 5(c) is implemented by following the approach proposed in [23] which designs a fully isolated minimal computing environment (FIMCE) on a multicore x86 platform. We develop a bare-metal ARM hypervisor running in the processor's Exception Level 2 (EL2) which is more privileged than the levels for the OS and applications. After launched on the Raspberry Pi board, the hypervisor configures the permission bits in the Stage-II translation table to block the OS and applications from accessing the serial port used by the fingerprint sensor. Hence, the adversary cannot access the fingerprint sensor to issue commands or steal fingerprint images. When available, a secure boot module can be used to assure that this configuration is properly set at boot time.

Upon receiving a request, the hypervisor creates a fully isolated computing environment consisting of a CPU core and a reserved physical memory region for the sensitive function to run. The CPU configuration ensures that maskable interrupts are not delivered the



(a) Fingerprint pre-processing.

(b) Hardware Setting

(c) Hypervisor Based $\mathcal{R}$oT. Arrows illustrate execution flow; shaded area denotes untrusted software.

Figure 5: Hardware and software components of our prototype.

core and non-maskable interrupts (NMIs) are trapped to the hypervisor. Thus, the untrusted OS cannot tamper with the environment via memory accesses or interrupts.

Running in the isolated environment is the code implementing $\mathcal{R}oT_P$ logic in the RTI protocol. Its signing key *sk* is stored in the hypervisor memory. To generate the response, it requests the hypervisor to run FV_{open} and $sign_{sk}$ in the FIMCE environment at runtime. The code of these two functions are self-contained without issuing system calls so that the executions do not depend on any untrusted code and data outside of the isolated environment. Considering that these two functions are for memory-resident computations without involving I/O operations, system calls are avoided by statically allocating the needed memory buffers. Note that an ARM CPU does not allow a user privilege code to issue hypercalls. Hence, we retrofit the OS with a special system call handler which issues the hypercall on behalf of $\mathcal{R}oT_P$.

5.2.3 Evaluation. Code complexity is shown in Table 2. We measured CPU execution time for FV_{OPEN} and $sign_{sk}$ within the virtualization-based $\mathcal{R}$oT and normal user space on the Raspberry PI board. Results are reported in Table 3. We note that time differences are not large.

System Component	LoC
ARM hypervisor	456 (C) and 906 (Assembly)
Self-contained FV_{OPEN}	701
Crypto library (incl. RSA and hash functions)	5,032

Table 2: Code Complexity (in LoC).

	FV_{OPEN}	$sign_{sk}$
Native environment	848.7	79.2
Virtualization-based $\mathcal{R}$oT	1143.51	75.6

Table 3: CPU time comparison. Average of out of 1000 executions (time in *ms*). Variance was negligible and omitted

In fact, the RSA signing operation has a slight performance advantage when running in the $\mathcal{R}$oT. The reason might be its exclusive use of the CPU core since interrupts are blocked.

6 PRACTICAL CONSIDERATIONS

We now discuss some practical issues relevant to the proposed RTI protocol.

6.1 Biometric Sensor Availability

One limitation of our general approach is the requirement for a biometric sensor hardwired to the $\mathcal{R}$oT. Our prototype shows how this requirement can be circumvented – the protocol can be securely deployed on devices not equipped with embedded biometric sensors by using a stand-alone biometric sensor and a trusted micro-hypervisor to emulate a hardware direct channel between the sensor and $\mathcal{R}oT_A$.

Nonetheless, we recognize that it might be beneficial to remove this hardware dependence. In particular, it would be interesting to develop new RTI protocols that use other types of physical challenges through other sensors that (similar to biometrics) are hard to clone/replay. In particular, developing alternative RTI based on other sensors that might be available on commodity devices and evaluating their usability trade-offs is an interesting future direction.

6.2 Biometric Confidentiality

One concern with the proposed protocol is confidentiality of the biometric data used in the protocol. Even though Dev-A might be compromised, the biometric sample is read directly by trusted $\mathcal{R}oT_A$. In other words, confidentiality of the user's biometric vis-a-vis Dev-A is guaranteed, assuming that $\mathcal{R}$oT hardware tamper-resistance is preserved. The same applies to $\mathcal{V}rf$, if it is also equipped with a $\mathcal{R}$oT. Otherwise, the owner of $\mathcal{V}rf$ should be the same as the user providing providing the biometric.

6.3 Fuzzy Extractor Issues

Statistical and reusability attacks are well-known issues of several FE constructions, including fuzzy vaults used in our prototype. The former is the biometric analog to dictionary attacks on passwords. It analyses the distribution of minutiae in human biometrics and uses this information to extract BT or $\mathcal{C}$hal from HD. The latter applies to non-reusable FEs. In such cases, obtaining two instances HD_1 and

HD_2, generated from the same biometric allows reconstruction of BT in clear.

We note that these attacks are a serious concern for FE-based biometric authentication where HD appears in clear. Whereas, in our case, the problem is obviated by transmitting HD over a secure channel to $\mathcal{R}oT_P$. In particular, we do not use FEs for biometric confidentiality (since they are not necessary to achieve that purpose). They are used such that $\mathcal{V}rf$ can always embed a fresh challenge $\mathcal{C}hal$ into the "biometric-based" challenge, preventing replays of previous RTI executions with the same biometric on other $\mathcal{R}oT$, e.g., $\mathcal{R}oT^*$.

6.4 Usability

As mentioned earlier, usability is a problem with sight-based presence attestation, along with its reliance on precise timing. Recall that location- and scene-based presence attestation schemes incur lower user burden. However, they also offer much lower security. Meanwhile, user burden in our protocol amounts to performing two biometric samplings: one with $\mathcal{V}rf$ and one with Dev-A. (Moreover, the user can pre-enroll his fingerprints with $\mathcal{V}rf$ well ahead of time.)This type of user interaction is common for authentication purposes and typically considered more convenient than other authentication means, such as entering a PIN or password. Therefore, we consider usability of biometric-based RTI protocol to be quite reasonable.

6.5 Accuracy

Accuracy of the underlying biometric matching is not affected by our use-case. Improving its accuracy is an orthogonal effort. Nonetheless, for completeness, we report on the accuracy considering the implementation used in our prototype. Similar accuracy analysis for biometric matching using fuzzy vaults (also considering other biometrics modalities) can be found in [14, 21, 24]. We report on our prototype's accuracy considering metrics for:

– **Genuine Acceptance Rate (GAR):** Percentage of biometric samples correctly matched to other samples acquired from the same biometric.

– **False Acceptance Rate (FAR):** Percentage of biometric samples incorrectly matched to any sample not acquired from the same biometric.

We conducted accuracy experiments using FVC2000 publicly available fingerprint database (database and further information available at: http://bias.csr.unibo.it/fvc2000/). FVC2000 includes multiple fingerprint images (10 different noisy images of each fingerprint) acquired using 4 types of low-cost biometric sensors. As discussed in Section 3.2, the FV polynomial degree allows configuring the number of matching data points in two biometric samples necessary to consider that the samples belong to the same user. Therefore, accuracy results are presented as a function of FV polynomial degree in Figure 6. According to the results in Figure 6, for a security-critical task such as RTI, an ideal choice would be degree 9 with nearly zero false acceptances. The same degree results in GAR of 80%, meaning that 1 out of 5 times a genuine RTI execution would fail and the user would need to try one more time.

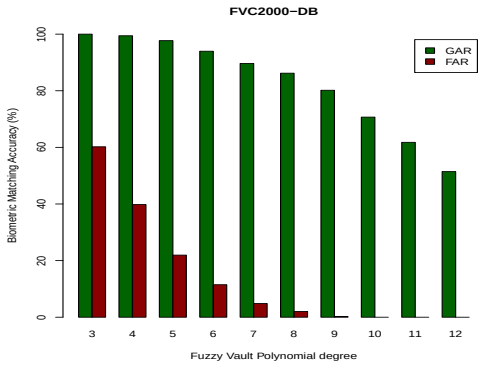


Figure 6: Accuracy of biometric matching in our prototype

.

7 RELATED WORK

In this section we summarize topics related to RTI, except *PA* [5] which was already discussed in Section 2.4.

Cuckoo Attacks were thoroughly introduced and formally modeled in [3]. Several potential solutions were analyzed under that model and, among them, secure hardware channels between Dev-A I/O interfaces the $\mathcal{R}oT_A$ were considered as the preferred method. As discussed in Section 2, even direct channels can be circumvented by a Cuckoo $\mathcal{A}dv$ that deploys its own *accomplice challenger* to replay $\mathcal{V}rf$ messages through the appropriate channel. To tackle this problem, our biometric-based approach explores the uniqueness of biometrics as a physical unclonable challenge, in addition to the existing secure channel between the biometric sensor and the $\mathcal{R}oT$.

Distance Bounding (DB) is a promising approach for addressing the RTI problem. With recent advances [25–27], DB could allow $\mathcal{V}rf$ to precisely establish maximum distance (bound) to the untrusted $\mathcal{R}oT_P$. Basically, if each device is equipped with DB facilities (a special radio and a high-precision clock) and $\mathcal{R}oT_P$ has a secure hardware channel to DB in its housing device, then the user can simply make sure that no other device is within the reported bound, e.g., 20-30 cm. However, several obstacles (discussed in Section 2) must be overcome before DB can be used for RTI.

User Trust Bootstrapping allows the user to establish trust on her device. TrustICE [28] uses a hardware approach and uses an LED under exclusive control of $\mathcal{R}oT$. The light signal emitted by this LED is used to convince the user that the device has an active $\mathcal{R}oT$. Other approaches [29, 30] reserve a fraction of $\mathcal{D}ev$ screen to communicate the state of the trusted component to the user. While these approaches succeed to communicate the state of $\mathcal{R}oT$ in a given device, they do not provide identification of corresponding public keys.

Device Pairing is the problem of initializing a secure (usually wireless) channel between two previously unfamiliar devices, without any trusted third party. Many device pairing protocols have been proposed, relying on various physical properties [31–33]. The main difference between RTI and device pairing is that, in the former, one of the devices (Dev-A) is potentially compromised and is therefore subject to cuckoo attacks. In contrast, device pairing mainly considers evil twin attacks.

Remote Attestation is an $\mathcal{R}$oT-enabled security service that allows $\mathcal{V}$rf to measure software state of applications running on $\mathcal{D}$ev. In recent years, several remote attestation techniques and architectures [34–39] were proposed, targeting different platforms and offering different types of guarantee. While remote attestation enables malware detection on a remote $\mathcal{D}$ev, it cannot be used as a means to solve RTI by ensuring that $\mathcal{D}$ev is in a malware-free state. This is because remote attestation itself requires mitigating the RTI problem, i.e., making sure that a remote attestation protocol indeed executes on $\mathcal{D}$ev before it can be used to ensure that $\mathcal{D}$ev is malware-free.

Biometrics are widely used in user authentication [2, 40–42] and identification [43, 44] systems. Fuzzy extractors are typically deployed to preserve biometric template confidentiality in the back-end of these systems [45]. To the best of our knowledge, this paper is the first proposal to use biometrics and fuzzy extractors to convey an unclonable challenge and assist in the identification of an $\mathcal{R}$oT.

8 CONCLUSION

This paper introduced and analyzed the RTI problem, which occurs whenever an $\mathcal{R}$oT is used to implement a security service that depends on physical IO devices (sensors and actuators) and relies on the assumption of $\mathcal{R}$oT residing in a specific physical device. To address this problem we proposed an RTI protocol based on the difficulty of cloning biometrics in real time. It uses the biometric as a challenge in the RTI protocol and relies on the existence of a hardware channel between biometric sensors and TEEs – a feature already available on some current devices. We also demonstrated a prototype implementation of our approach.

ACKNOWLEDGMENTS

We thank IPSN'21 anonymous referees for their helpful comments. This research was supported in part by funding from Army Research Office (ARO) contract W911NF-16-1-0536, Semiconductor Research Corporation (SRC) contract 2019-TS-2907, as well as NSF Awards 1956393 (SATC) and 1840197 (CICI). This article was also partially supported by the Singapore National Research Foundation under NCR Award Number NRF2018NCR-NSOE004-0001.

REFERENCES

[1] S. Mirzamohammadi, J. A. Chen, A. A. Sani, S. Mehrotra, and G. Tsudik, "Ditio: Trustworthy auditing of sensor activities in mobile & iot devices," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, p. 28, ACM, 2017.

[2] S. Srinivas, D. Balfanz, E. Tiffany, F. Alliance, and A. Czeskis, "Universal 2nd factor (u2f) overview," *FIDO Alliance Proposed Standard*, pp. 1–5, 2015.

[3] B. Parno, J. M. McCune, and A. Perrig, *Bootstrapping trust in modern computers*. Springer Science & Business Media, 2011.

[4] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *VLSI Design*, 2004.

[5] Z. Zhang, X. Ding, G. Tsudik, J. Cui, and Z. Li, "Presence attestation: The missing link in dynamic trust bootstrapping," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 89–102, ACM, 2017.

[6] D. P. Shepard, T. E. Humphreys, and A. A. Fansler, "Evaluation of the vulnerability of phasor measurement units to gps spoofing attacks," *International Journal of Critical Infrastructure Protection*, vol. 5, no. 3-4, pp. 146–153, 2012.

[7] M. L. Psiaki, T. E. Humphreys, and B. Stauffer, "Attackers can spoof navigation signals without our knowledge. here's how to fight back gps lies," *IEEE Spectrum*, vol. 53, no. 8, pp. 26–53, 2016.

[8] S. Brands and D. Chaum, "Distance-bounding protocols," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 344–359, Springer, 1993.

[9] G. P. Hancke and M. G. Kuhn, "An rfid distance bounding protocol," in *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pp. 67–73, IEEE, 2005.

[10] C. H. Kim and G. Avoine, "Rfid distance bounding protocol with mixed challenges to prevent relay attacks," in *International Conference on Cryptology And Network Security*, pp. 119–133, Springer, 2009.

[11] S. Capkun, K. El Defrawy, and G. Tsudik, "Group distance bounding protocols," pp. 302–312, 2011.

[12] K. B. Rasmussen and S. Capkun, "Realization of rf distance bounding.," in *USENIX Security Symposium*, pp. 389–402, 2010.

[13] A. Dhar, E. Puddu, K. Kostiainen, and S. Capkun, "Proximitee: Hardened sgx attestation by proximity verification." Cryptology ePrint Archive, Report 2018/902, 2018. https://eprint.iacr.org/2018/902.

[14] I. D. O. Nunes, K. Eldefrawy, and T. Lepoint, "Snuse: A secure computation approach for large-scale user re-enrollment in biometric authentication systems," *Future Generation Computer Systems*, 2019.

[15] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*, pp. 523–540, Springer, 2004.

[16] A. Juels and M. Sudan, "A fuzzy vault scheme," *Des. Codes Cryptography*, vol. 38, pp. 237–257, Feb. 2006.

[17] A. Kiayias and M. Yung, "Cryptographic hardness based on the decoding of reed-solomon codes," in *International Colloquium on Automata, Languages, and Programming*, pp. 232–243, Springer, 2002.

[18] "Android fingerprint hardware interface." https://source.android.com/security/authentication/fingerprint-hal (accessed 2019-09-07).

[19] K. Ko, "User's guide to nist biometric image software (nbis)," *NIST Interagency/Internal Report (NISTIR)-7392*, 2007.

[20] V. Shoup, "Ntl: A library for doing number theory," *www.shoup.net/ntl/*, 2001.

[21] K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE transactions on information forensics and security*, vol. 2, no. 4, pp. 744–757, 2007.

[22] B. Tams, "Absolute fingerprint pre-alignment in minutiae-based cryptosystems," in *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)*, pp. 1–12, Sept 2013.

[23] S. Zhao and X. Ding, "On the effectiveness of virtualization based memory isolation on multicore platforms," in *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, (Washington, DC, USA), IEEE Computer Society, 2017.

[24] Y. J. Lee, K. Bae, S. J. Lee, K. R. Park, and J. Kim, "Biometric key binding: Fuzzy vault based on iris images," in *International Conference on Biometrics*, pp. 800–808, Springer, 2007.

[25] P. Leu, M. Singh, and S. Capkun, "Message time of arrival codes: A fundamental primitive for secure distance measurement," 2020.

[26] M. Singh, P. Leu, and S. Capkun, "UWB with pulse reordering: Securing ranging against relay and physical-layer attacks," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, 2019.

[27] M. Singh, P. Leu, A. Abdou, and S. Capkun, "Uwb-ed: Distance enlargement attack detection in ultra-wideband," in *28th USENIX Security Symposium (USENIX Security 19)*, (Santa Clara, CA), pp. 73–88, USENIX Association, Aug. 2019.

[28] H. Sun, K. Sun, Y. Wang, J. Jing, and H. Wang, "Trustice: Hardware-assisted isolated computing environments on mobile devices," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 367–378, IEEE, 2015.

[29] J. Danisevskis, M. Peter, J. Nordholz, M. Petschick, and J. Vetter, "Graphical user interface for virtualized mobile handsets," *IEEE S&P MoST*, 2015.

[30] M. Lange and S. Liebergeld, "Crossover: secure and usable user interface for mobile devices with multiple isolated os personalities," in *Proceedings of the 29th Annual Computer Security Applications Conference*, pp. 249–257, ACM, 2013.

[31] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pp. 6–pp, IEEE, 2006.

[32] C. Soriente, G. Tsudik, and E. Uzun, "Hapadep: human-assisted pure audio device pairing," in *International Conference on Information Security*, pp. 385–400, Springer, 2008.

[33] E. K. Jung, M. A. Malamud, A. J. Cohen, R. W. Lord, R. A. Levien, and J. D. Rinaldo Jr, "Device pairing via device to device contact," Apr. 12 2011. US Patent 7,925,022.

[34] V. Haldar, D. Chandra, and M. Franz, "Semantic remote attestation: a virtual machine directed approach to trusted computing," in *USENIX Virtual Machine Research and Technology Symposium*, vol. 2004, 2004.

[35] M. Barbosa, B. Portela, G. Scerri, and B. Warinschi, "Foundations of hardware-based attested computation and application to sgx," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 245–260, IEEE, 2016.

[36] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust," in *NDSS*, vol. 12, pp. 1–15, 2012.

[37] K. Eldefrawy, N. Rattanavipanon, and G. Tsudik, "Hydra: hybrid design for remote attestation (using a formally verified microkernel)," in *Proceedings of the 10th ACM Conference on Security and Privacy in wireless and Mobile Networks*, pp. 99–110, ACM, 2017.

[38] I. D. O. Nunes, K. Eldefrawy, N. Rattanavipanon, M. Steiner, and G. Tsudik, "VRASED: A verified hardware/software co-design for remote attestation," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1429–1446, 2019.

[39] M. Ammar, B. Crispo, and G. Tsudik, "Simple: A remote attestation approach for resource-constrained iot devices," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 247–258, IEEE, 2020.

[40] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain, "Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 450–455, 2005.

[41] P. M. Burger, "Biometric authentication system," 2001. US Patent 6,219,439.

[42] R. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor, and M. Savvides, "Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption," 2015.

[43] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb, "Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7905–7916, 2015.

[44] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *2013 Proceedings IEEE INFOCOM*, pp. 2652–2660, IEEE, 2013.

[45] I. D. O. Nunes, K. Eldefrawy, and T. Lepoint, "Secure non-interactive user re-enrollment in biometrics-based identification and authentication systems," in *International Symposium on Cyber Security Cryptography and Machine Learning*, pp. 162–180, Springer, 2018.