Towards Heterogeneous Remote Attestation Protocols

Paul Georg Wagner[1] and Jürgen Beyerer[1,2]
[1]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[2]*Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany*

Keywords: Remote Attestation, Trusted Computing, Trusted Platform Modules, Intel SGX, Arm Trustzone, Distributed Systems.

Abstract: Remote attestation protocols are valuable tools to cryptographically verify the integrity of remote software stacks. Usually these protocols rely on a specific hardware-based trusted computing technology to provide their security guarantees. However, especially in distributed settings with many collaborating platforms it is not always feasible to use protocols developed exclusively for one trusted computing technology. In this work we explore the possibility of conducting heterogeneous remote attestations between endpoints utilizing different trusted computing technologies. We motivate the benefits of such attestations in the light of distributed systems and present a list of requirements for a working heterogeneous remote attestation protocol. Then we propose a remote attestation mechanism that can securely link Intel SGX enclaves, TPM-based trusted applications, as well as ARM TrustZone devices with an attested and encrypted communication channel. Finally we outline how this mechanism can be integrated into an established remote attestation protocol.

1 INTRODUCTION

In today's environment of ubiquitous computing, more and more data are being processed in distributed systems. Such systems consist of components that are often operated by different stakeholders in a decentralized infrastructure. One major challenge in such scenarios is to establish the trustworthiness of remote system components. This includes verifying the correctness of remote components and preventing malicious operators from manipulating critical software. A popular way of achieving this is by applying *remote attestation* protocols. Remote attestation is the process of cryptographically verifying the integrity of a computing device's software stack. This verification process is usually backed by hardware-based security mechanisms known as *trusted computing* technologies. Currently the most popular and widespread trusted computing technologies are Trusted Platform Modules (TPMs), Intel SGX and ARM TrustZone.

While much research went into the development of these technologies over recent years, so far they have mainly been viewed separately from each other. Depending on the specific use case and security requirements, usually only one technology is selected and deployed systemwide. However, especially in distributed systems it is beneficial to combine several trusted computing technologies and conduct remote attestations between them. That way technologies with different benefits and drawbacks can be used simultaneously. For example, TPMs are cheap and easy to use, but generally do not provide as strong security guarantees as Intel SGX. Being able to deploy SGX enclaves for security-critical components while still using TPM-protection would greatly enhance the flexibility and adaptability of distributed trusted applications. Furthermore, in large distributed systems it is unlikely that all operators unanimously agree on a single trusted computing technology to use in the entire system. Finally, there are often technological reasons for using multiple trusted computing technologies. For example, ARM-based embedded devices seldom have a dedicated TPM module, so they are bound to rely on the ARM TrustZone technology.

In this position paper we take first steps towards a heterogeneous remote attestation protocol that connects different trusted computing technologies. For this, we first give a brief overview of existing remote attestation protocols (section 2), before we identify requirements for a heterogeneous protocol (section 3). Then we describe how to conduct remote attestations between TPMs, Intel SGX and ARM TrustZone and show how these mechanisms can be integrated into a joint attestation protocol that establishes secure channels (section 4). Finally we conclude the paper with an outlook on future work (section 5).

Wagner, P. and Beyerer, J.
Towards Heterogeneous Remote Attestation Protocols.
DOI: 10.5220/0011289000003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 586-591
ISBN: 978-989-758-590-6; ISSN: 2184-7711
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

2 RELATED WORK

Among the most widespread trusted computing technologies today are Trusted Platform Modules (TPMs). TPMs hold several cryptographic keys on a dedicated hardware chip, which can then be used to encrypt and sign critical information in a secure environment. The private parts of the cryptographic keys stored in the TPM hardware are protected against external influence and cannot be extracted in plain text. There have been multiple proposals for TPM-based remote attestation protocols over the years (Armknecht et al., 2008; Zhou and Zhang, 2010; Wagner et al., 2020; TCG, 2019a). All of them utilize a special set of registers in the TPM chip, the Platform Configuration Registers (PCRs). When a TPM-protected system boots up, these registers contain unforgeable fingerprints of the system's current hardware and software configuration. During the remote attestation protocol, the PCR contents are signed with the private part of an asymmetric attestation key, which is known only to the TPM. By verifying that signature as well as the system fingerprints, remote verifiers can be convinced that the attested system runs a correctly configured and unmodified software stack. Furthermore, most remote attestation protocols also conduct a Diffie-Hellman key exchange to establish a shared secret between verifier and prover.

A more sophisticated trusted computing technology is Intel's Software Guard Extension (SGX). Intel SGX consists of a set of special hardware security modules directly integrated into the CPU die, which allow security critical code to run as a protected enclave (Costan and Devadas, 2016). SGX enclaves are executed completely isolated from the rest of the computer system, including other system processes, administrators, and even the operating system itself. Similar to TPMs, there are several SGX-based remote attestation protocols on offer. One way of remotely attesting SGX enclaves is by using a group signature scheme called EPID (Johnson et al., 2016). This scheme extends the machine-internal local attestation of an SGX enclave with a cryptographic signature that is externally verifiable by communicating with Intel's attestation services. However, with recent CPUs Intel also provides the option to build decentralized third-party attestation infrastructures (DCAP), mainly targeted to data center operations (Scarlata et al., 2018). Both attestation protocols offer a sophisticated Sigma protocol encompassing a Diffie-Hellman key exchange to establish a shared secret between prover and verifier. The low-level remote attestation schemes provided by Intel have also been integrated into several higher-level communication protocols. There are proposals for including SGX-based remote attestation into TLS (Knauth et al., 2018) as well as HTTPS (King and Wang, 2021). Furthermore, Google uses a high-level protocol called EKEP[1] for SGX-based attestation in their Asylo framework, and there is also a protocol implementation for Gramine[2].

While Intel SGX is only available for x86 architectures, TrustZone is the trusted computing technology for ARM-based infrastructures. ARM TrustZone divides the system into two distinct worlds, the normal world and the secure world (Pinto and Santos, 2019). Both worlds have their own software stack and are strictly isolated from each other by means of hardware-based access control. That way the potentially compromised normal world (and even its operating system) cannot maliciously influence security critical applications running inside the secure world. Unlike with TPMs and SGX, remote attestation capabilities have not been a big concern in the design of ARM TrustZone. To our knowledge, no standardized remote attestation protocol for TrustZone has been published so far. However, there are some application-specific proposals (Shepherd et al., 2017; Wang et al., 2020; Ling et al., 2021) available. These proposals utilize a Software-TPM implemented as a trusted application and protected by the TrustZone hardware. Hence the resulting proposals are very similar to TPM-based attestation protocols.

3 PROTOCOL REQUIREMENTS

As a first step towards heterogeneous remote attestation protocols, we have identified six general requirements that such protocols should fulfill.

Heterogeneous Attestations. The first and most important requirement is the support of heterogeneous remote attestations. This means that a single protocol instance must be able to deal with different trusted computing technologies on both endpoints. After the protocol handshake completes successfully, the code identities of both sides have to be mutually verified as dictated by the respective mechanisms. Furthermore, at this point the protocol has to consider the potentially different attestation scopes. For example, an SGX remote attestation provides code identity information about a single enclave. TPMs on the other hand always include the entire trusted system in their attestation data. In the remainder of this paper we focus on supporting TPMs, SGX and TrustZone

[1] https://asylo.dev/docs/concepts/ekep.html
[2] https://gramineproject.io/

as trusted computing technologies. However, in the future other technologies should be supported in heterogeneous remote attestations as well.

Mechanism Negotiation. Since a heterogeneous remote attestation protocol has to handle multiple different technologies at once, a suitable way of mechanism negotiation is required. The protocol needs to be able to select suitable attestation mechanisms for both respective endpoints and notify them about the kind of attestation information to expect from the other side. Only then can a heterogeneous attestation endpoint successfully validate the transmitted attestation information. This could be achieved with a separate negotiation phase before the actual remote attestation.

Secure Channels. Besides the mutual verification of code integrity, the remote attestation protocol also has to establish a secure channel between the attested endpoints. Usually this is achieved by adopting a Diffie-Hellman key exchange to establish a shared secret on the channel. However, when developing a heterogeneous remote attestation protocol, great care has to be taken to properly authenticate this key exchange. The key exchange has to be unambiguously linked to the attested code identities. This means that the shared secret must only be accessible to the attested endpoints, and only if they are in the correct (i.e. verified) states. How exactly this is achieved depends on the specific trusted computing technology used. Furthermore, the protocol must be resilient against replay attacks, i.e. prevent adversaries from authenticating a new key exchange with old attestation information.

Platform Independence. A heterogeneous remote attestation protocol is particularly useful in distributed systems consisting of many different hardware platforms, because in such systems it is often not feasible to deploy just one single trusted computing technology. Hence platform independence is an important protocol requirement. This includes compatible protocol implementations for different platforms (e.g. ARM and x86), but also simple protocol bindings for different programming languages. Furthermore the protocol should be easily expandable with novel trusted computing mechanisms.

Protocol Reuse. Since there are sophisticated and (partly) standardized remote attestation protocols available for the considered trusted computing technologies, it is not necessary to invent new attestation mechanisms completely from scratch. Instead, the heterogeneous attestation protocol should build on already existing attestation mechanisms wherever possible. That way protocol complexity can be decreased and compatibility can be maximized.

Performance. Finally, the protocol performance is defined by the time it takes to conduct a full two-way handshake including the mutual attestations. However, the protocol performance will strongly depend on the specific trusted computing technologies. For example, TPMs are rather slow on certain cryptographic operations, while SGX enclaves can utilize the full resources of the CPU and are much faster. Hence the performance of a heterogeneous remote attestation protocol should be evaluated by comparing it with existing attestation protocols of the various mechanisms.

4 A HETEROGENEOUS REMOTE ATTESTATION PROTOCOL

In this section we show how mutual remote attestation can be conducted between endpoints of different trusted computing technologies. We present a concept for mutual remote attestation between hardware-based TPMs and Intel SGX enclaves, and discuss how it can be adopted for attestations between ARM TrustZone devices and SGX enclaves. We also show how secure communication channels can be established between heterogeneous endpoints. Finally, we propose the integration of these mechanisms into a heterogeneous remote attestation protocol.

4.1 Connecting Trusted Platform Modules with SGX Enclaves

To facilitate a mutual remote attestation between hardware-based TPMs and Intel SGX enclaves, we combine the `EREPORT` instruction on SGX-capable systems with the `TPM2_Quote` instruction on the TPM-protected side. Furthermore, we establish a shared secret using an elliptic curve Diffie-Hellman (ECDH) key exchange. We choose ECDH to allow the use of the `TPM2_ECDH_ZGen` key establishment primitive of the TPM 2.0 interface (TCG, 2019b), as has been recently proposed in (Wagner et al., 2020). On the SGX side, this key exchange is authenticated by including a hash of the public key in the signed `EREPORT` data structure. Together, this establishes a secure communication channel between the heterogeneous endpoints and ensures that the shared secret is known only to both attested code identities.

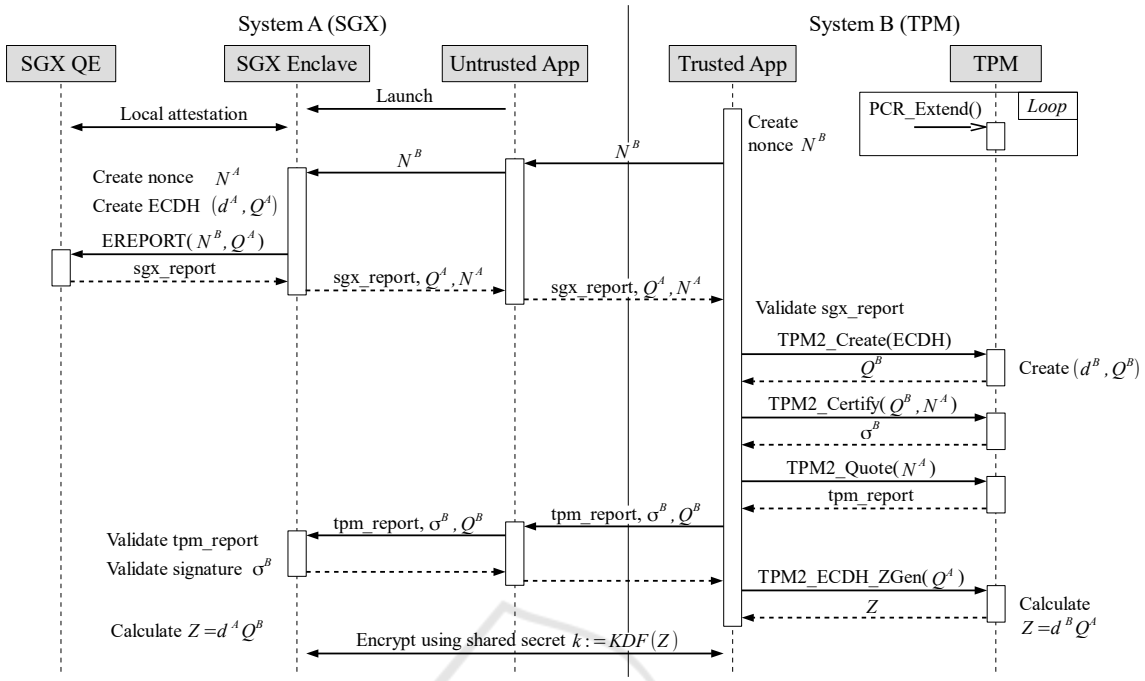

Figure 1: Remote attestation mechanism between SGX enclaves and TPMs.

Figure 1 shows a high-level description of the proposed heterogeneous remote attestation sequence between an SGX enclave and a TPM-protected trusted application. To keep the sequence diagram simple, we assume that both systems are already set up and ready to conduct a remote attestation. This means that the TPM-protected system has performed a trusted boot sequence that established a chain of trust from the hardware TPM over the bootloader and operating system to the user applications. We especially assume that the trusted application has been included in the measurements conducted via the `PCR_Extend` TPM operation, because otherwise the SGX enclave cannot validate the remote code identity and the attestation would fail. On the SGX side, we assume that the quoting enclave (QE) has been launched and provisioned with an attestation key according to either the EPID or DCAP specification. Note that for simplicity, figure 1 only provides a high-level description of the necessary SGX attestation messages. The details of EPID (Johnson et al., 2016) and DCAP (Scarlata et al., 2018) attestation messages can be found in the respective specifications. Furthermore we omit additional SGX components required for the respective attestation procedure, such as the Intel attestation services (with EPID) or the provisioning certification enclave (with DCAP). As first step in the protocol handshake, the TPM-protected trusted application randomly draws a nonce N^B and transmits it to the remote enclave. This nonce is used for freshness purposes and prevents malicious reuse of attestation information (replay attacks). The SGX enclave then creates its own nonce N^A for the same reason, as well as a new ECDH key pair (d^A, Q^A). Note that the secret key d^A has been created inside the secure SGX enclave and will never leave it. The public key Q^A along with the remote nonce N^B is then used as qualifying data to generate the signed SGX report in the quoting enclave. This means that the tuple (Q^A, N^B) is hashed and included in the attestation information generated for the SGX enclave. That way the nonce provides freshness, and the enclave's ECDH public key is authenticated and linked to the conducted attestation. The resulting SGX attestation report is then transmitted back to the TPM system, alongside the enclave's public key Q^A and its nonce N^A. In the trusted application, the SGX attestation report can then be verified. During the verification process, both the report's signature as well as the contained enclave fingerprints have to be validated. Note that figure 1 omits some details of the signature verification step. When the EPID signature scheme is used on the SGX side, the verifier has to communicate with Intel's attestation service (IAS). When DCAP is used, a third-party certification authority provides the necessary certificates for the signature verification. Once the SGX attestation report has passed verification, the TPM-protected software stack generates its ECDH key pair and attestation information. As described in (Wagner et al., 2020), we can use the `TPM2_Create` function

to generate a new ECDH key pair (d^B, Q^B) inside the TPM. Once again, the private key d^B will never leave the protected TPM hardware for the remainder of the handshake. Instead, we use the `TPM2_Certify` function to sign the public ECDH key Q^B with the current attestation key used by the TPM. While this differs from the key handling on the SGX side, it fulfills the same purpose. The resulting signature σ^B authenticates the ECDH public key and links it to the current attestation context, since it is signed by the unique attestation key. Note that to prevent replay attacks, here we have to include the remote nonce N^A in the signature as well. Finally, the usual `TPM2_Quote` function is used to generate an attestation report over the selected PCRs, which is also signed with the TPM's attestation key. As soon as the TPM attestation report, the ECDH public key and its signature is transmitted to the SGX enclave, it can be verified there. In this verification step the enclave has to check both the report's and the public key's signatures and also validate the attested PCR fingerprints. Once this is successful, both sides can calculate the shared Diffie-Hellman secret Z using their own secret key and the authenticated remote public key. Subsequent communication can then be encrypted using a symmetric secret derived from Z.

4.2 Connecting ARM TrustZone Devices with SGX Enclaves

To give our heterogeneous remote attestation protocol a broader field of application, we aim to support ARM TrustZone devices without access to a physical TPM. One issue is that to our knowledge there is currently no standardized remote attestation protocol available for ARM TrustZone. However, we can build on existing proposals that are using a dedicated trusted measurer application protected by the TrustZone hardware (Shepherd et al., 2017). For our purposes it makes sense to use fTPM (Raj et al., 2016) as trusted measurer. fTPM is a TPM 2.0 reference implementation created by Microsoft, which has been migrated to ARM TrustZone devices. As a software implementation of the TPM specification, it offers all functionality of dedicated hardware TPM modules. Using fTPM-based remote attestation on ARM TrustZone has clear benefits for our use case. Since fTPM provides a TPM 2.0 compatible programming interface, we can very easily adopt the protocol presented in figure 1 for TrustZone devices. The only real difference in the protocol sequence is that we use an fTPM trusted application instead of a hardware TPM. However, there are also several drawbacks of this solution. First, software TPMs cannot provide the same level of isolation as a dedicated hardware TPM chip. In fact, ARM TrustZone only provides hardware-backed isolation of trusted software against malicious applications running in the rich world. However, depending on the attacker model we require protection against maliciously manipulated trusted applications as well. Furthermore, it is still unclear how to establish a broad chain of trust that includes all relevant system software (including the bootloader and operating system) from both worlds. This is an important step to ensure the integrity of the fTPM trusted application. There is some recent work by ARM aimed at establishing measured boot processes on ARM platforms with fTPM[3], but to our knowledge this largely remains an open issue. However, besides the drawbacks of this approach, we believe that fTPM-based remote attestation on ARM TrustZone devices is a promising direction for developing heterogeneous remote attestation in distributed environments containing ARM devices.

4.3 Heterogeneous Protocol Integration

After describing heterogeneous remote attestation mechanisms that are capable of establishing secure communication channels, we are left with integrating these mechanisms into a joint protocol. We plan to do this by extending the existing Enclave Key Exchange Protocol (EKEP)[4]. EKEP has been developed by Google for their Asylo trusted computing framework, where it is used to securely communicate between local and remote SGX enclaves. On a technical level, EKEP is based on a TLS variant that features mutual authentication, which has been enhanced to include attestation information during the key establishment. Integrating the heterogeneous remote attestation mechanism as described in figure 1 into EKEP requires some modifications. First, establishing the encrypted communication channel should now be outsourced to the underlying TLS protocol, instead of performing a separate Diffie-Hellman key exchange. However, to prevent man-in-the-middle attacks, the established channel encryption key still needs to be properly linked to the attested code identities. Usually this is achieved by generating self-signed TLS certificates. These TLS certificates are then included in the attestation reports instead of the ECDH public key itself (c.f. figure 1). This authenticates the self-signed certificates, which in turn can be used to finally establish an encrypted TLS channel during the remote attestation protocol handshake. While this is a simple and reliable solution, integrating the TPM-based attestation step shown in figure 1 requires some deeper modifications. This is because the key ex-

[3] https://github.com/OP-TEE/optee_os/pull/5025
[4] https://asylo.dev/docs/concepts/ekep.html

change should use the TPM to generate and authenticate the Diffie-Hellman key pair, which the native TLS integration in EKEP does not support yet. Furthermore, the most important requirement remaining to be addressed is mechanism negotiation. Fortunately EKEP already includes a way to signal attestation mechanisms to the remote endpoints. This is because EKEP has been designed to accommodate both local and remote attestations between SGX enclaves. While naturally these two attestation mechanisms strongly depend on each other, we are confident that this mechanism can be adopted to negotiate mechanisms between endpoints that are utilizing completely different trusted computing technologies.

5 CONCLUSION

In this position paper we addressed the challenge of conducting remote attestations between different trusted computing platforms. We motivated the usefulness of a heterogeneous remote attestation protocol that bridges the technological gap between different technologies and presented a list of protocol requirements. Then we showed how remote attestations can be conducted between Intel SGX enclaves and hardware-based TPMs, as well as ARM TrustZone devices. Our proposed attestation mechanisms also establish shared secrets that are bound to the attested platform identities. Finally, we briefly discussed the possibility of integrating the proposed mechanisms into the existing remote attestation protocol EKEP.

In the future, we plan to develop mechanisms to facilitate remote attestations with more trusted computing platforms, such as RISC-V and AMD SEV. Furthermore, important future work includes the implementation and evaluation of a proof-of-concept attestation protocol. As outlined in the last section, we plan to use the existing implementation of EKEP as a basis for this. We believe that we can fulfill most of the requirements presented in section 3 that way. However, even with a working proof of concept for multiple technologies, there will still be remaining issues to consider. This includes performance evaluations, but most importantly the problem of analyzing the security of heterogeneous attestation protocols. Since such protocols depend on multiple security-critical technologies with possibly different attacker models at once, determining the resulting security guarantees that can be expected from a protocol handshake is rather difficult. We plan to explore these research questions more thoroughly in the future.

REFERENCES

Armknecht, F., Gasmi, Y., Sadeghi, A.-R., Stewin, P., Unger, M., Ramunno, G., and Vernizzi, D. (2008). An efficient implementation of trusted channels based on openssl. In *3rd ACM workshop on Scalable trusted computing*, pages 41–50.

Costan, V. and Devadas, S. (2016). Intel sgx explained. *IACR Cryptology Archive*, page 86.

Johnson, S., Scarlata, V., Rozas, C., Brickell, E., and Mckee, F. (2016). Intel software guard extensions: Epid provisioning and attestation services. Technical report.

King, G. and Wang, H. (2021). Httpa: Https attestable protocol. *arXiv preprint arXiv:2110.07954*.

Knauth, T., Steiner, M., Chakrabarti, S., Lei, L., Xing, C., and Vij, M. (2018). Integrating remote attestation with transport layer security. *arXiv preprint arXiv:1801.05863*.

Ling, Z., Yan, H., Shao, X., Luo, J., Xu, Y., Pearson, B., and Fu, X. (2021). Secure boot, trusted boot and remote attestation for arm trustzone-based iot nodes. *Journal of Systems Architecture*, 119:102240.

Pinto, S. and Santos, N. (2019). Demystifying arm trustzone: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 51(6):1–36.

Raj, H., Saroiu, S., Wolman, A., Aigner, R., Cox, J., England, P., Fenner, C., Kinshumann, K., Loeser, J., Mattoon, D., et al. (2016). ftpm: A software-only implementation of a tpm chip. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 841–856.

Scarlata, V., Johnson, S., Beaney, J., and Zmijewski, P. (2018). Supporting third party attestation for intel sgx with intel data center attestation primitives. Technical report.

Shepherd, C., Akram, R. N., and Markantonakis, K. (2017). Establishing mutually trusted channels for remote sensing devices with trusted execution environments. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–10.

TCG (2019a). Trusted attestation protocol (tap) information model. Technical report. Accessed April 2022.

TCG (2019b). Trusted platform module 2.0 library. Technical report. Accessed April 2022.

Wagner, P. G., Birnstill, P., and Beyerer, J. (2020). Establishing secure communication channels using remote attestation with tpm 2.0. In *International Workshop on Security and Trust Management*, pages 73–89. Springer.

Wang, Z., Zhuang, Y., and Yan, Z. (2020). Tz-mras: A remote attestation scheme for the mobile terminal based on arm trustzone. *Security and Communication Networks*, 2020.

Zhou, L. and Zhang, Z. (2010). Trusted channels with password-based authentication and tpm-based attestation. In *2010 International Conference on Communications and Mobile Computing*, volume 1, pages 223–227. IEEE.