The Computer for the 21st Century: Security & Privacy Challenges After 25 Years

Leonardo B. Oliveira[1], Fernando Magno Quintão Pereira[1], Rafael Misoczki[2], Diego F. Aranha[3], Fábio Borges[4], Jie Liu[5]

Abstract— **Decades went by since Mark Weiser published his influential work on how a computer of the 21st century would look like. Over the years, some of the UbiComp features presented in that paper have been gradually adopted by industry players in the technology market. While this technological evolution resulted in many benefits to our society, it has also posed, along the way, countless challenges that we have yet to surpass. In this paper, we address major challenges from two areas that most afflict the UbiComp revolution: security and privacy. We examine open problems on software protection, long-term security, cryptography engineering, and privacy implications. We also point out promising directions towards the solutions of those problems. We claim that if we get all this right, we will turn the science fiction of UbiComp into science fact.**

I. Introduction

In 1991, Mark Weiser described a vision of the *Computer for the 21st Century* [1]. Weiser, in his prophetic paper, argued the most far-reaching technologies are those that allow themselves to disappear, vanish into thin air. According to Weiser, this oblivion is a human–not a technological–phenomenon: "Whenever people learn something sufficiently well, they cease to be aware of it," he claimed. This event is called "tacit dimension" or "compiling" and can be witnessed, for instance, when drivers react to street signs without consciously having to process the letters S-T-O-P [1].

A quarter of a century later, however, Weiser's dream is far from becoming true. Over the years, many of his concepts regarding Ubiquitous Computing (UbiComp) [2], [3] have been materialized into what today we call Wireless Sensor Networks [4], [5], Internet of Things [6], [7], Wearables [8], [9], and Cyber-Physical Systems [10], [11]. The applications of these systems range from traffic accident and CO_2 emission monitoring to autonomous automobile and patient in-home care. Nevertheless, besides all their benefits, the advent of those systems per se have also brought about some drawbacks. And, unless we address them appropriately, the continuity of Weiser's prophecy will be at stake.

UbiComp poses new drawbacks because, vis-à-vis traditional computing, it exhibits an entirely different outlook [12]. Computer systems in UbiComp, for instance, feature sensors, CPU, and actuators. Respectively, this means they can hear (or spy on) you, process your data (and, possibly, find out something confidential about you), and respond to your actions (or, ultimately, expose you by revealing some secret). Those capabilities, in turn, make proposals for conventional computers ill-suited in the UbiComp setting and present new challenges.

In the above scenarios, some of the most critical challenges lie in the areas of Security and Privacy [13]. This is so because the market and users often pursue a system full of features at the expense of proper operation and protection; although, conversely, as computing elements pervade our daily lives, the demand for stronger security schemes becomes greater than ever. Notably, there is a dire need for a secure mechanism able to encompass all aspects and manifestations of Ubicomp, across time as well as space, and in a seamless and efficient manner.

In this paper, we discuss contemporary security and privacy issues in the context of UbiComp. We examine multiple research problems still open and point to promising approaches towards their solutions. More precisely, we investigate the following challenges and their ramifications.

1) Software protection in Section II: we study the impact of the adoption of weakly typed languages by resource-constrained devices, and discuss mechanisms to mitigate this impact. We go over techniques to validate polyglot software (i.e., software based on multiple programming languages), and revisit promising methods to analyze networked embedded systems.
2) Long-term security in Section III: we examine the security of today's widely used cryptosystems (e.g. RSA/ECC-based), present some of the latest threats (e.g. the advances in cryptanalysis and quantum attacks), and explore new directions and challenges to guarantee long-term security in the UbiComp setting.
3) Cryptography engineering in Section IV: we restate the essential role of cryptography in safeguarding computers, discuss the status quo of lightweight cryptosystems and their secure implementation, and highlight challenges in key management protocols.
4) Privacy implications in Section V: we explain why security is necessary but not sufficient to ensure privacy, go over important privacy-related issues (e.g. sensitivity data identification and regulation), and discuss some tools of the trade to fix those (e.g. privacy-preserving protocols based on homomorphic encryption).

We claim that only if we get the challenges right, we can turn the science fiction of UbiComp into science fact.

*This work is partially supported by FAPEMIG, CNPq, and CAPES.

[1]UFMG, Brazil {leob,fernando} at dcc.ufmg.br
[2]Intel Labs rafael.misoczki at intel.com
[3]Unicamp, Brazil dfaranha at ic.unicamp.br
[4]LNCC, Brazil borges at lncc.br
[5]Microsoft Research jie liu at microsoft.com

II. Software Protection

Modern UbiComp systems are rarely built from scratch. Components developed by different organizations, with different programming models and tools, and under different assumptions are integrated to offer complex capabilities. To this end, we analyze the software ecosystem that characterizes the field. Figure 1 provides a high-level representation of this ecosystem. We focus specially on three aspects of this environment, which pose security challenges to developers: the security shortcomings of C and C++, the dominant programming languages among cyber-physical implementations; the interactions between these languages and other programming languages, and the consequences of these interactions on the distributed nature of UbiComp applications. We start by diving deeper into the idiosyncrasies of C and C++.

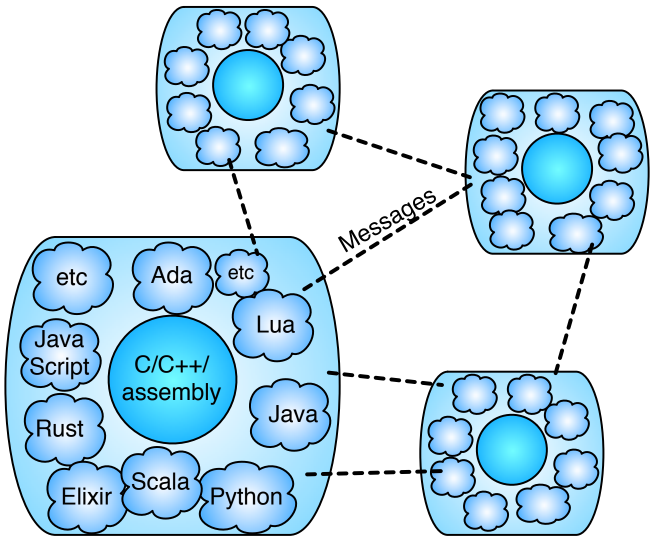


Fig. 1: A UbiComp System is formed by modules implemented as a combination of different programming languages. This diversity poses challenges to software security.

A. Type Safety

A great deal of the software used in UbiComp systems is implemented in C or in C++. This fact is natural, given the unparalleled efficiency of these two programming languages. However, if C/C++ are efficient, on the one hand, their weak type system gives origin to a plethora of software vulnerabilities. In programming language's argot, we say that a type system is weak when it does not support two key properties: *progress* and *preservation* [14]. The formal definitions of these properties is immaterial for the discussion that follows. It suffices to know that, as a consequence of weak typing, neither C, nor C++, ensure, for instance, bounded memory accesses. Therefore, programs written in these languages can access invalid memory positions. As an illustration of the dangers incurred by this possibility, it suffices to know that out-of-bounds access are the principle behind buffer overflow exploits.

The software security community has been developing different techniques to deal with the intrinsic vulnerabilities of C/C++/assembly software. Such techniques can be fully static, fully dynamic or a hybrid of both approaches. Static protection mechanisms are implemented at the compiler level; dynamic mechanisms are implemented at the runtime level. In the rest of this section, we list the most well-known elements in each category.

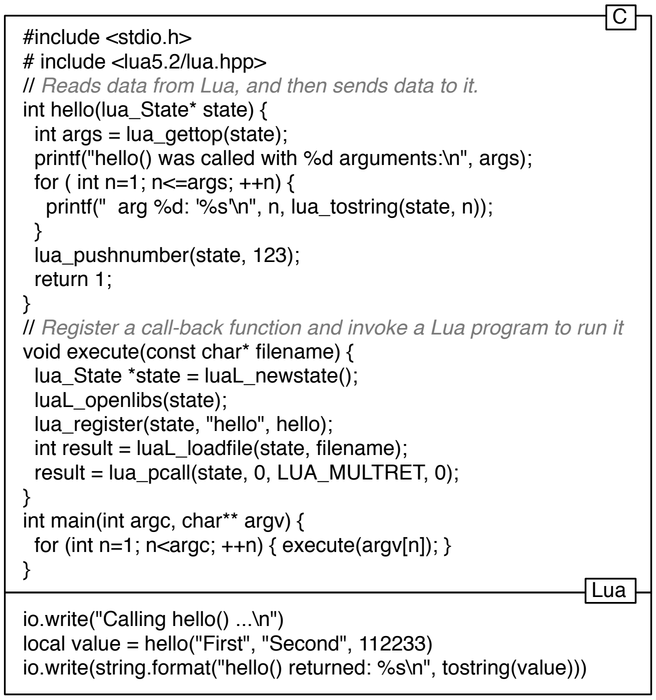
Static analyses provide a conservative estimate of the program behavior, without requiring the execution of such a program. This broad family of techniques includes, for instance, *abstract interpretation* [15], *model checking* [**?**] and *guided proofs* [16]. The main advantage of static analyses is the low runtime overhead, and its soundness: inferred properties are guaranteed to always hold true. However, static analyses have also disadvantages. In particular, most of the interesting properties of programs lay on undecidable land [17]. Furthermore, the verification of many formal properties, even though a decidable problem, incur a prohibitive computational cost [18].

Dynamic analyses come in several flavors: testing (KLEE [19]), profiling (Aprof [20], Gprof [21]), symbolic execution (DART [22]), emulation (Valgrind [23]), and binary instrumentation (Pin [24]). The virtues and limitations of dynamic analyses are exactly the opposite of those found in static techniques. Dynamic analyses usually do not raise false alarms: bugs are described by examples, which normally lead to consistent reproduction [25]. However, they are not required to always find security vulnerabilities in software. Furthermore, the runtime overhead of dynamic analyses still makes it prohibitive to deploy them into production software [26].

As a middle point, several research groups have proposed ways to combine static and dynamic analyses, producing different kinds of hybrid approaches to secure low-level code. This combination might yield security guarantees that are strictly more powerful than what could be obtained by either the static or the dynamic approaches, when used separately [27]. Nevertheless, negative results still hold: if an attacker can take control of the program, usually he or she can circumvent state-of-the-art hybrid protection mechanisms, such as control flow integrity [28]. This fact is, ultimately, a consequence of the weak type system adopted by languages normally seen in the implementation of UbiComp systems. Therefore, the design and deployment of techniques that can guard such programming languages, without compromising their efficiency to the point where they will no longer be adequate to UbiComp development, remains an open problem.

B. Polyglot Programming

Polyglot programming is the art and discipline of writing source code that involves two or more programming languages. It is common among implementations of cyber-physical systems. As an example, Ginga, the Brazilian protocol for digital TV, is mostly implemented in Lua and C [29]. Figure 2 shows an example of communication between a C and a Lua program. Other examples of interactions between programming languages include bindings between C and



```c
#include <stdio.h>
# include <lua5.2/lua.hpp>
// Reads data from Lua, and then sends data to it.
int hello(lua_State* state) {
  int args = lua_gettop(state);
  printf("hello() was called with %d arguments:\n", args);
  for ( int n=1; n<=args; ++n) {
    printf("  arg %d: '%s'\n", n, lua_tostring(state, n));
  }
  lua_pushnumber(state, 123);
  return 1;
}
// Register a call-back function and invoke a Lua program to run it
void execute(const char* filename) {
  lua_State *state = luaL_newstate();
  luaL_openlibs(state);
  lua_register(state, "hello", hello);
  int result = luaL_loadfile(state, filename);
  result = lua_pcall(state, 0, LUA_MULTRET, 0);
}
int main(int argc, char** argv) {
  for (int n=1; n<argc; ++n) { execute(argv[n]); }
}
```

```lua
io.write("Calling hello() ...\n")
local value = hello("First", "Second", 112233)
io.write(string.format("hello() returned: %s\n", tostring(value)))
```

Fig. 2: Two-way communication between a C and a Lua program.

Python [30], C and Elixir [31] and the Java Native Interface [32]. Polyglot programming complicates the protection of systems. Difficulties arise due to a lack of multi-language tools and due to unchecked memory bindings between C/C++ and other languages.

An obstacle to the validation of polyglot software is the lack of tools that analyze source code written in different programming languages, under a unified framework. Returning to Figure 2, we have a system formed by two programs, written in different programming languages. Any tool that analysis this system as a whole must be able to parse these two distinct syntaxes, and infer the connection points between them. Work has been performed towards this end, but solutions are still very preliminary. As an example, Maas *et al.* [30] have implemented automatic ways to check if C arrays are correctly read by Python programs. As another example, Furr and Foster [33] have described techniques to ensure type-safety of OCaml-to-C and Java-to-C bindings.

A promising direction to analyze polyglot systems is based on the idea of compilation of source code partially available. This feat consists in the reconstruction of the missing syntax and the missing declarations necessary to produce a minimal version of the original program that can be analyzed by typical tools. The analysis of code partially available makes it possible to test parts of a polyglot program in separate, in a way to produce a cohesive view of the entire system. This technique has been demonstrated to yield analyzable Java source code, for instance [34]. Notice that this type of reconstruction is not restricted to high-level programming languages. Testimony of this fact is the notion of *micro execution*, introduced by Patrice Godefroid [35]. Godefroid's tool allows the testing of x86 binaries, even when object files are missing. Nevertheless, in spite of these developments, the reconstruction is still restricted to the static semantics of programs. The synthesis of behavior is a thriving discipline in computer science [36], but still far away from enabling the certification of polyglot systems.

C. Distributed Programming

Ubiquitous computing systems tend to be distributed. It is even difficult to conceive any use for an application in this world that does not interact with other programs. And, it is common knowledge that distributed programming opens up several doors to malicious users. Therefore, to make cyber-physical technology safer, security tools must be aware of the distributed nature of such systems. Yet, two main challenges stand in front of this requirement: the difficulty to build a holistic view of the distributed application, and the lack of semantic information bound to messages exchanged between processes that communicate through a network.

To be accurate, the analysis of a distributed system needs to account for the interactions between the several program parts that constitute this system [37]. Discovering such interactions is difficult, even if we restrict ourselves to code written in a single programming language. Difficulties stem from a lack of semantic information associated with operations that send and receive messages. In other words, such operations are defined as part of a library, not as part of the programming language itself. Notwithstanding this fact, there are several techniques that infer communication channels between different pieces of source code. As examples, we have the algorithms of Greg Bronevetsky [38], and Teixeira *et al.* [39], which build a distributed view of a program's control flow graph (CFG). Classic static analyses work without further modification on this *distributed* CFG. However, the distributed CFG is still a conservative approximation of the program behavior. Thus, it forces already imprecise static analyses to deal with communication channels that might never exist during the execution of the program.

Tools that perform automatic analyses in programs rely on static information to produce more precise results. In this sense, types are core for the understanding of programs. For instance, in Java, and other object-oriented programming languages, the type of objects determine how information flows along the program code. However, despite this importance, messages exchanged in the vast majority of distributed systems are not typed. Reason for this is the fact that such messages, at least in C, C++ and assembly software, are arrays of bytes. To mitigate this problem, researchers have devised analyses that infer the contents [40] and the size of arrays [41] in weakly-typed programming languages. Yet, many problems along this direction remain open. For instance, analyzing arrays in low-level programming languages often requires the ability to deal with pointer arithmetic. Pointer analysis that handle arithmetic still struggle to deal with real-world applications [42]. Therefore, the design of tools able to analyze the very fabric of UbiComp systems is still a problem to be solved.

III. Long-term Security

Various UbiComp systems are designed to withstand a lifespan of many years, even decades [43], [44]. For example, systems in the context of critical infrastructure may require an enormous financial investment to be designed and deployed in the field. Therefore, they would offer a better return-over-investment if they remain in use for longer time.

The automotive area is another field of particular interest. Vehicles are expected to be functional and reliable for several years, even decades. Renewing vehicle fleets and even updating certain of their features (*recalls*) translate into less profit. They can be seen as part of the UbiComp ecosystem as they are equipped with several embedded devices, already offer connectivity to the internet and future enriched driving experiences will heavily rely on the data wirelessly collected and shared across vehicles and the infrastructure [45].

It is also worth mentioning that systems designed to endure a lifespan of several years or decades might be subject to lack of future maintenance. The competition among players able to innovate is very aggressive leading to a high rate of companies going out of business within a few years [46]. A world inundate by devices whose no one is accountable for their proper operation would likely offer serious future challenges.

From the aforementioned examples, and among many other systems, it is evident that there is an increasing interest for UbiComp systems to be reliable for a longer time and, whenever possible, requiring as fewer updates as possible. These requirements have a straightforward impact on the security features of such systems: they would offer fewer opportunities for patching eventual security breaches than conventional systems. This is a critical situation given the intense and dynamic research field of devising and exploiting new security breaches. Therefore, it is of utmost importance to understand what are the scientific challenges to ensure long-term security from the early stage of the system design, instead of resorting to palliative measures a posteriori.

A. Cryptography as The Core Component

Ensuring long-term security is a quite challenging task for any system, not only for UbiComp systems. At a minimum, it requires that every single security component is future-proof per-se and in face of the interconnections between the various components. The main ingredient in most security mechanisms is Cryptography, as highlighted in section IV, and therefore that will be the focus of our assessment.

Traditional cryptographic techniques rely on the hardness of computational problems such as integer factorization [47] and discrete logarithm problems [48]. These problems are believed to be intractable by current cryptanalysis techniques and technological resources; therefore cryptographers were able to build secure instantiation of cryptosystems. For various reasons (to be discussed in our assessment) however, the future-proof condition of such schemes is at stake.

B. Advancements in Classical Cryptanalysis

The first threat for the future-proof condition of any cryptosystem refers to potential advancements on cryptanalysis, i.e., on the techniques aiming at solving the underlying security problem in a more efficient way (processing time, memory and others) than originally predicted. Widely-deployed schemes have a history of academic and industrial scrutiny and, thus, one would expect little or no progress on the cryptanalysis techniques of those schemes. Yet, the literature has recently shown some interesting and unexpected results.

In [49], for example, Barbulescu et al. introduced a new quasi-polynomial algorithm to solve the discrete logarithm problem in finite fields of small characteristic. The discrete logarithm problem is the underlying security problem of the Diffie-Hellman Key Exchange [50], the Digital Signature Algorithm [51] and their elliptic curve variants (ECDH [52] and ECDSA [51], respectively), just to mention a few widely-deployed cryptosystems. This cryptanalytic result is restricted to finite fields of small characteristic, what represents an important limitation to attack real-world deployments of the aforementioned schemes. However, any sub-exponential algorithm that solves a longstanding problem can be seen as a relevant indication that the cryptanalysis literature is still subject to eventual breakthroughs.

This situation should be considered by architects designing UbiComp systems that have longterm security as a requirement. Implementations that support increased security levels are preferred when compared to fixed, single key size support, for instance. In this way, UbiComp systems could consciously accommodate future cryptanalytic advancements or, at the very least, reduce the costs for security upgrades.

C. Future Disruption Due to Quantum Attacks

Quantum computers are expected to offer dramatic speedups to solve certain computational problems, as enlightened by Daniel R. Simon in his seminal paper on quantum algorithms [53]. To our misfortune, some of the affected computational problems are currently being used to secure widely-deployed cryptosystems.

As an example, Lov K. Grover introduced a quantum algorithm [54] able to find an element in the domain of a function (of size N) which leads with high probability to a sought output in only $O(\sqrt{N})$ steps. This algorithm is used to speed up the cryptanalysis of symmetric cryptography. Block ciphers of n bits keys, for example, would offer only $n/2$ bits of security against a quantum adversary. Hash functions would be affected in different ways, depending on the expected security property. Hash functions of n bits digests would offer only $n/3$ bits of security against collision attacks and $n/2$ bits of security against pre-image attacks. In this context, AES-128 and SHA-256 collision-resistance would not meet acceptable security level. Note that both block ciphers and hash function constructions will still remain secure if longer key and digest sizes are employed. However, this would bring about important performance challenges. AES-256, for example, is 40% less efficient than AES-128 (due to the 14 rounds, instead of 10).

Even more critical than the scenario for symmetric cryptography, quantum computers will offer an exponential speedup to attack most of the widely-deployed public-key cryptosystems. This is due to Peter Shor's algorithm [55] which can efficiently factor large integers and compute the discrete logarithm of an element in large groups. The impact of this work will be devastating to RSA and ECC-based schemes, as increasing the key sizes would not suffice: they will need to be completely replaced.

In the field of quantum resistant public-key cryptosystems, several challenges need to be addressed. The first one refers to establishing a consensus in both academia and industry on the recommended quantum-resistant replacements. For example, there are two main techniques able to withstand quantum attacks, namely: post-quantum cryptography (PQC) and quantum cryptography. The former is based on classical computational problems believed to be hard even for quantum computers. PQC schemes can be implemented in current computers [56], [57]. The latter depends on quantum infrastructure to be deployed and is mostly used for key-exchange purposes [58]. The limited availability and high costs of quantum infrastructure deployment favor the increasing interest in the post-quantum cryptography trend.

Among the PQC schemes, there are cryptosystems based on different computational problems. Hash-based signatures are the most accredited solutions for digital signatures. These schemes [59], [60] are improvements on top of Merkle scheme [61] and their security relies solely on certain well-known properties of hash functions (thus secure against quantum attacks, assuming appropriate digest sizes are used). Other security features, however, such as key exchange and asymmetric encryption, have not reached a consensus.

Regarding standardization efforts, NIST has recently started a standardization process for PQC [62] which should take at least a few more years to be concluded. This pendency might represent a challenge per se given forthcoming systems that will need to remain secure for a long timeframe. In this sense, deploying non-standardized post-quantum schemes can be seen as interim manner to provide quantum resistance.

Another challenge in the context of post-quantum public-key cryptosystems refers to their new implementation requirements. As mentioned before, hash-based signatures are very promising post-quantum candidates (given efficiency and security reduction to hash functions) but also bring about a new set of implementation challenges, such as the task of keeping the signature state secure. In short, these schemes have private-keys that evolve along the time. If rigid state management policies are not in place, a signer can re-utilize the same private-key twice which would void the security guarantees offered by the scheme. Recently, initial works to address these new implementation challenges have appeared in the literature [63]. A recent construction [64] showed how to get rid of the state management issue at the price of fairly larger signatures. These examples indicate that new cryptosystems will result in new implementation challenges that must be addressed by UbiComp systems architects.

IV. Cryptography Engineering

UbiComp systems involve building blocks of very different natures: hardware components such as sensors and actuators, embedded software implementing communication protocols and interface with cloud providers, and ultimately operational procedures and other human factors. As a result, pervasive systems have a large attack surface that must be protected using a combination of techniques.

Cryptography is a fundamental part of any modern computing system, but unlikely to be the weakest component in its attack surface. Networking protocols, input parsing routines and even interface code with cryptographic mechanisms are components much more likely to be vulnerable to exploitation. However, a successful attack on cryptographic security properties can be usually disastrous. For example, violations of confidentiality may cause massive data breaches involving sensitive information. Adversarial interference on communication integrity may allow command injection attacks that deviate from the specified behavior. Availability is crucial to keep the system accessible by legitimate users and to guarantee continuous service provisioning, thus cryptographic mechanisms must also be lightweight to minimize potential for abuse by attackers.

Physical access by adversaries to portions of the attack surface is a particularly challenging aspect of deploying cryptography in UbiComp systems. By assumption, adversaries can recover long-term secrets and credentials that provide some control over a (hopefully small) portion of the system. Below we will explore some of the main challenges in deploying cryptographic mechanisms for pervasive systems, including how to manage keys and realize efficient and secure implementation of cryptography.

A. Key management

UbiComp systems are by definition heterogeneous platforms, connecting devices of massively different computation and storage power. Designing a cryptographic architecture for any heterogeneous system requires assigning clearly defined roles and security properties to each entity in the system. Resource-constrained devices should receive less computationally intensive tasks, and their lack of tamper-resistance protections indicate that long-term secrets should not reside in these devices. More critical tasks involving expensive public-key cryptography should be delegated to more powerful nodes. A careful trade-off between security properties, functionality and cryptographic primitives must then be addressed per device or class of devices [65], following a set of guidelines for pervasive systems.

1) Efficiency: protocols should be lightweight and easy to implement, mandating that traditional public key infrastructures (PKIs) and expensive certificate handling operations are restricted to the more powerful and connected nodes in the architecture. Alternative models supporting implicit certification include identity-based [66] (IBC) and certificateless cryptography [67] (CLPKC), the former implying inherent key escrow. The difficulties with key revocation still impose obstacles for their wide adoption, despite progress [68].

2) Functionality: key management protocols must manage lifetime of cryptographic keys and ensure accessibility to the currently authorized users, but handling key management and authorization separately may increase complexity and vulnerabilities. A promising way of combining the two services into a cryptographically-enforced access control framework is attribute-based encryption [69], [70], where keys have sets of capabilities and attributes that can be authorized and revoked on demand.

3) Communication: components should minimize the amount of communication, at risk of being unable to operate if communication is disrupted. Non-interactive approaches for key distribution [71] are recommended here, but advanced protocols based on bilinear pairings should be avoided due to recent advances on solving the discrete log problem (in the so called medium prime case [72]). These advances forcedly increase the parameter sizes and reduce performance/scalability, favoring more traditional forms of asymmetric cryptography.

4) Interoperability: pervasive systems are composed of components originating from different manufacturers. Supporting a cross-domain authentication and authorization framework is crucial for interoperability [73].

Cryptographic primitives involved in joint functionality must then be compatible with all endpoints and respect the constraints of the less powerful devices.

B. Lightweight cryptography

The emergence of huge collections of interconnected devices in UbiComp motivate the development of novel cryptographic primitives, under the moniker *lightweight cryptography*. The term lightweight does not imply *weaker* cryptography, but application-tailored cryptography that is especially designed to be efficient in terms of resource consumption such as processor cycles, energy and memory footprint [74]. Lightweight designs aim to target common security requirements for cryptography, but may adopt less conservative choices or more recent building blocks.

As a first example, many new block ciphers were proposed as lightweight alternatives to the Advanced Encryption Standard (AES) [75]. Important constructions are LS-Designs [76], modern ARX and Feistel networks [77], and substitution-permutation networks [78], [79]. A notable candidate is the PRESENT block cipher, after a 10-year maturity of resisting cryptanalytic attempts [80].

In the case of hash functions, a design may even trade-off advanced security properties (such as collision resistance) for simplicity in some scenarios. A clear case is the construction of short Message Authentication Codes (MAC) from non-collision resistant hash functions, such as in SipHash [81], or digital signatures from short-input hash functions [82]. In conventional applications, BLAKE2 [83] is a stronger drop-in replacement to recently cryptanalyzed standards [84].

Another trend is to provide confidentiality and authentication in a single step, through Authenticated Encryption with Associated Data (AEAD). This can be implemented with a block cipher operation mode (like GCM [85]) or a dedicated design. The CAESAR competition is expected to select a new AEAD algorithm for standardization later this year.

In terms of public-key cryptography, Elliptic Curve Cryptography (ECC) [48], [86] continues to be the main contender in the space against factoring-based cryptosystems [47], due to an underlying problem conjectured to be fully exponential. Modern instantiations of ECC enjoy high performance and implementation simplicity and are very suited for embedded systems [87], [88], [89]. The dominance of number-theoretic primitives is however threatened by quantum computers as described in Section III.

The plethora of new primitives must be rigorously evaluated from both the security and performance point of views, involving both theoretical work and engineering aspects. Implementations are expected to consume smaller amounts of energy [90], cycles and memory [91] in ever decreasing devices and under more invasive attacks.

C. Side-channel resistance

If implemented without care, an otherwise secure cryptographic algorithm or protocol can leak critical information which may be useful to an attacker. Side-channel attacks [92] are a significant threat against cryptography and may use timing information, cache latency, power and electromagnetic emanations to recover secret material. These attacks emerge from the interaction between the implementation and underlying computer architecture and represent an intrinsic security problem to pervasive computing environments, since the attacker is assumed to have physical access to at least some of the legitimate devices.

Protecting against intrusive side-channel attacks is a challenging research problem, and countermeasures typically promote some degree of *regularity* in computation. *Isochronous* or constant time implementations were among the first strategies to tackle this problem in the case of variances in execution time or latency in the memory hierarchy. Application of formal methods has developed the first tools to verify isochronicity of implementations, such as information flow analysis [93] and program transformations [94].

While there is a recent trend towards constructing and standardizing cryptographic algorithms with some embedded resistance against the simpler timing and power analysis attacks [76], more powerful attacks such as differential power analysis [95] or fault attacks [96] are very hard to prevent or mitigate. Fault injection became a much more powerful attack methodology after demonstrated in software [97].

Masking techniques [98] are frequently investigated as a countermeasure to decorrelate leaked information from secret data, but frequently require robust entropy sources to achieve their goal. Randomness recycling techniques have been useful as a heuristic, but formal security analysis of such approaches is an open problem [99]. Modifications in the underlying architecture in terms of instruction set extensions, simplified execution environments and transactional mechanisms for restarting faulty computation are another promising research direction, but may involve radical and possibly cost-prohibitive changes to current hardware.

V. Privacy implications

Security–or, notably, its property confidentiality–is a necessary but not sufficient condition for a protocol to ensure privacy [100]. To ensure security, UbiComp systems must implement protection mechanisms like secure communication channels, in a way that only authorized senders encrypt messages for authorized receivers to decrypt. To ensure privacy, it is important to determine the relevant sources of leakage and employ privacy-preserving protocols to manipulate sensitive data. Such protocols often need to use cryptosystems to guarantee that the receiver can obtain the required information without inferring sensitive data about the sender.

A. Application Scenario Challenges

When senders and receivers are considered in the context of a particular application scenario, there are two major challenges. The first is the identification of sensitive data, as it might differ from culture to culture. The second is how to deal with multiple potentially conflicting regulations.

1) Identifying Sensitive Data: Deciding what may be sensitive data might be a challenging task. The article 12 of the Universal Declaration of Human Rights proclaimed by the United Nations General Assembly in Paris on 10 December 1948 states: *No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks.* In general, more attention is needed to data that enable monitoring products, animals, and people. Such data can be used for behavioral profiling to enable prediction and manipulation. Practical privacy-preserving protocols enable the receiver to obtain consolidated information or to collect data aggregated by means of addition or concatenation, anonymously. For instance, location data raises several privacy issues and, thus, few people share their location [101], although smartphones can disclose more sensitive data than location [102]. UbiComp systems for healthcare process sensitive data [103], and security cameras can leak sensitive data [104]. The privacy challenges have not only to do with citizens, but also industries [105]. Even if a message is encrypted, the metadata–i.e., network traffic–can reveal sensitive information [106].

2) Regulation: The regulations about privacy are budding around the world, and laws vary from a country to another. It is a challenge for international institutions developing UbiComp systems to market them around the world in compliance with the law. Another problem can be the absence of law, enabling competitors to develop strategies regarding private information that increase market advantage over more ethical corporations. In general, violations of privacy might change an election and might lead to a surveillance society [107]. Independent of application scenarios, privacy-preserving protocols face their own technological challenges.

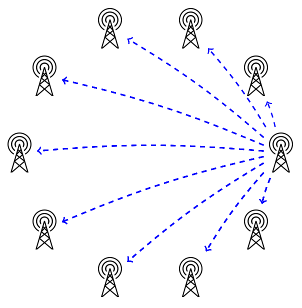
B. Technological Challenges

1) Computing all operators: In theory, any operator can be computed over encrypted data [108], such techniques are known as fully homomorphic encryption. In practice, it is a challenge to construct a fully homomorphic encryption for many application scenarios. Researchers usually develop privacy-preserving protocols based on additive homomorphic encryption [109], [110] and DC-Nets (from "Dining Cryptographers") [111]. Both techniques compute the addition operator over encrypted data. However, the former are functions, and the latter are families of functions. Given an additive homomorphic encryption, we can construct an asymmetric DC-Net [111].

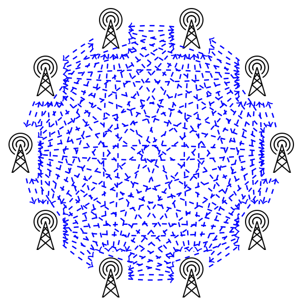
2) Trade-off between enforcement and malleability: DC-Nets can enforce privacy ensuring that no one can decrypt messages unless all encrypted messages were taken into account. Homomorphic encryption does not enforce privacy, i.e., single messages can be decrypted. This happens because DC-Nets are non-malleable and homomorphic encryption is malleable. DC-Net schemes, on the one hand, enable attackers to change encrypted messages and use other encrypted messages to imply the value of others in the aggregation process. On the other hand, it is easier to generate and to distribute keys for homomorphic encryption schemes than for DC-Net schemes.

3) Key distribution: Using homomorphic encryption, the receiver just needs to generate a public-private key pair and transmit the public key to the senders in an authenticated way, which will share the same key to encrypt their messages. Although messages might have the same content, encrypted messages will be different from each other because homomorphic encryption schemes are probabilistic. Using DC-Net schemes, a protocol does not need a receiver because the senders can be the receivers. For this reason, let us just call them participants, which generate their private keys. For symmetric DC-Nets, each participant shares two keys with each other, i.e., sends a secret to and receives another secret from each other. Afterward, each participant has a private key given by the list of secrets. Hence, each participant encrypts a message $m_{i,j}$ using the function $\mathfrak{M}_{i,j} \leftarrow \mathrm{Enc}(m_{i,j}) = m_{i,j} + \sum_{o \in \mathcal{M}-\{i\}} \mathrm{Hash}(s_{i,o} \mid\mid j) - \mathrm{Hash}(s_{o,i} \mid\mid j)$, where $m_{i,j}$ is the message sent by the participant i in the time j, Hash is a secure hash function predefined by the participants, $s_{i,o}$ is the secret sent from participant i to participant o, similarly, $s_{o,i}$ is the secret received by i from o, and $||$ is the concatenation operator. Each participant i can sent the encrypted message $\mathfrak{M}_{i,j}$ to each other. Hence, they can decrypt the aggregated encrypted messages computing $\mathrm{Dec} = \sum_{i \in \mathcal{M}} \mathfrak{M}_{i,j} = \sum_{i \in \mathcal{M}} m_{i,j}$. No one can decrypt if one or more messages are missing. For asymmetric DC-Nets, each participant generates a private key used as encryption key, and then, they add the keys using a homomorphic encryption or a symmetric DC-Net to generate the decryption key, which can be made public.

Notice that homomorphic encryption schemes tend to have low overhead for setting up keys and for distributing



(a) Homomorphic encryption.



(b) Symmetric DC-Nets.

Fig. 3: Setting up the keys.

them. The receiver of the aggregated encrypted messages just generates a public-private key pair and sends the public key to each sender. Symmetric DC-Nets need $O(I^2)$ messages to set up the keys, where I is the number of participants. Figure 3 depicts the messages to set up keys using (a) homomorphic encryption and (b) symmetric DC-Nets.

4) Aggregation and Consolidation: Homomorphic encryption schemes cannot transmit encrypted messages directly to the receiver because individual messages can be decrypted. Senders should aggregate the encrypted messages somehow before the receiver gets the encrypted aggregation. DC-Net schemes can send encrypted messages directly to the receiver–and all participants–because DC-Nets can enforce privacy. The aggregation for homomorphic encryption is challenging while it is trivial for DC-Nets due to the trade-off in section V-B.2. Notice that we are referencing DC-Nets as fully connected DC-Nets, aggregation for other DC-Nets is based on trust, which is also challenging. A step further is the consolidation, which generates more information than the aggregation. Text or alphanumeric messages are good examples. Consolidating messages keeping users' privacy is usually more common and challenging than aggregation.

5) Performance: Most commitment schemes, asymmetric DC-Nets, and homomorphic encryption use modular multi-exponentiations, which can be computed efficiently [112]. Symmetric DC-Net schemes need an iteration over the number of participants. Thus, their processing time increases as this number increases becoming the most expensive to compute the encryption. This number is not relevant to schemes based on modular multi-exponentiations, which are deemed efficient for traditional computers, but still inefficient for constrained devices.

6) Failures and Disruption: Eventually, UbiComp systems may fail, and users might become attackers and then disrupt systems. We have commitment and asymmetric DC-Net schemes to deal with such events. However, designing protocols that detect and prevent such events is challenging. Usually, failures and disruption imply two problems with the messages: incorrect contents and interruptions. The former invalidates or blurs the aggregation. Commitment and asymmetric DC-Net schemes can audit encrypted messages. The latter preclude the aggregation for DC-Net schemes. Only homomorphic encryption can decrypt an aggregation without considering all encrypted messages.

VI. Conclusion

In the words of Mark Weiser, Ubiquitous Computing is "the idea of integrating computers seamlessly into the world at large" [1]. Thus, far from being a phenomenon from this time, the design and practice of UbiComp systems were already being discussed one quarter of a century ago. In this paper, we have revisited this notion, which permeates the most varied levels of our society, under a security and privacy point of view. In the coming years, these two topics will occupy much of the time of researchers and engineers. In our opinion, the use of this time should be guided by a few observations. First, UbiComp software is often produced as the combination of different programming languages, sharing a common core implemented in a type-unsafe language such as C, C++ or assembly. Second, physical access and constrained resources complicate the design of efficient and secure cryptographic algorithms, which are often amenable to side-channel attacks. Third, the availability and volume of data often manipulated by UbiComp systems complicates the protection of users' privacy. Fourth, the life span of such systems, coupled with the difficult to update and re-deploy them, makes them vulnerable to the inexorable progress of technology, which brings new players, such as post-quantum cryptography. Given these observations, and the importance of ubiquitous computing, it is easy to conclude that the future holds fascinating challenges waiting for the attention of the academia and the industry.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[2] ——, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[3] K. Lyytinen and Y. Yoo, "Ubiquitous computing," *Communications of the ACM*, vol. 45, no. 12, pp. 63–96, 2002.

[4] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *MobiCom'99*, 1999, pp. 263–270.

[5] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[6] K. Ashton, "That 'Internet of Things' Thing," *RFiD Journal*, vol. 22, pp. 97–114, 2009.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] S. Mann, "Wearable computing: A first step toward personal imaging," *Computer*, vol. 30, no. 2, pp. 25–32, 1997.

[9] T. Martin and J. Healey, "2006's wearable computing advances and fashions," *IEEE Pervasive Computing*, vol. 6, no. 1, 2007.

[10] E. A. Lee, "Cyber-physical systems-are computing foundations adequate," in *NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, vol. 2. Citeseer, 2006.

[11] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *47th Design Automation Conference*. ACM, 2010.

[12] G. D. Abowd and E. D. Mynatt, "Charting past, present, and future research in ubiquitous computing," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, no. 1, pp. 29–58, 2000.

[13] F. Stajano, *Security for Ubiquitous Computing*. John Wiley and Sons, 2002.

[14] B. C. Pierce, *Types and Programming Languages*, 1st ed. The MIT Press, 2002.

[15] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *POPL*. New York, NY, USA: ACM, 1977, pp. 238–252.

[16] X. Leroy, "Formal verification of a realistic compiler," *Commun. ACM*, vol. 52, no. 7, pp. 107–115, 2009.

[17] H. G. Rice, "Classes of recursively enumerable sets and their decision problems," *Trans. Amer. Math. Soc.*, vol. 74, no. 1, pp. 358–366, 1953.

[18] R. P. Wilson and M. S. Lam, "Efficient context-sensitive pointer analysis for c programs," in *PLDI*. New York, NY, USA: ACM, 1995, pp. 1–12.

[19] C. Cadar, D. Dunbar, and D. Engler, "KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs," in *OSDI*. USENIX, 2008, pp. 209–224.

[20] E. Coppa, C. Demetrescu, and I. Finocchi, "Input-sensitive profiling," in *PLDI*. New York, NY, USA: ACM, 2012, pp. 89–98.

[21] S. L. Graham, P. B. Kessler, and M. K. McKusick, "gprof: a call graph execution profiler (with retrospective)," in *Best of PLDI*. New York, NY, USA: ACM, 1982, pp. 49–57.

[22] P. Godefroid, N. Klarlund, and K. Sen, "Dart: directed automated random testing," in *PLDI*. New York, NY, USA: ACM, 2005, pp. 213–223.

[23] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," in *PLDI*. New York, NY, USA: ACM, 2007, pp. 89–100.

[24] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *PLDI*. New York, NY, USA: ACM, 2005, pp. 190–200.

[25] A. A. Rimsa, M. D'Amorim, and F. M. Q. Pereira, "Tainted flow analysis on e-SSA-form programs," in *CC*. Springer, 2011, pp. 124–143.

[26] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "Addresssanitizer: a fast address sanity checker," in *ATC*. USENIX, 2012, pp. 28–28.

[27] A. Russo and A. Sabelfeld, "Dynamic vs. static flow-sensitive security analysis," in *CSF*. Washington, DC, USA: IEEE, 2010, pp. 186–199.

[28] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "Control-flow bending: On the effectiveness of control-flow integrity," in *SEC*. Berkeley, CA, USA: USENIX, 2015, pp. 161–176.

[29] L. F. G. Soares, R. F. Rodrigues, and M. F. Moreno, "Ginga-NCL: the declarative environment of the brazilian digital tv system," *J. Braz. Comp. Soc*, vol. 12, no. 4, pp. 1–10, 2007.

[30] A. J. Maas, H. Nazaré, and B. Liblit, "Array length inference for c library bindings," in *ASE*. New York, NY, USA: ACM, 2016, pp. 461–471.

[31] G. Fedrecheski, L. C. P. Costa, and M. K. Zuffo, "Elixir programming language evaluation for iot," in *ISCE*. Washington, DC, USA: IEEE, 2016, p. Online.

[32] J. S. Rellermeyer, M. Duller, K. Gilmer, D. Maragkos, D. Papageorgiou, and G. Alonso, "The software fabric for the internet of things," in *IOT*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 87–104.

[33] M. Furr and J. S. Foster, "Checking type safety of foreign function calls," *ACM Trans. Program. Lang. Syst.*, vol. 30, no. 4, pp. 18:1–18:63, 2008.

[34] B. Dagenais and L. Hendren, "Enabling static analysis for partial java programs," in *OOPSLA*. New York, NY, USA: ACM, 2008, pp. 313–328.

[35] P. Godefroid, "Micro execution," in *ICSE*. New York, NY, USA: ACM, 2014, pp. 539–549.

[36] Z. Manna and R. J. Waldinger, "Toward automatic program synthesis," *Commun. ACM*, vol. 14, no. 3, pp. 151–165, 1971.

[37] H. A. López, E. R. B. Marques, F. Martins, N. Ng, C. Santos, V. T. Vasconcelos, and N. Yoshida, "Protocol-based verification of message-passing parallel programs," in *OOPSLA*. New York, NY, USA: ACM, 2015, pp. 280–298.

[38] G. Bronevetsky, "Communication-sensitive static dataflow for parallel message passing applications," in *CGO*. Washington, DC, USA: IEEE, 2009, pp. 1–12.

[39] F. A. Teixeira, G. V. Machado, F. M. Q. Pereira, H. C. Wong, J. M. S. Nogueira, and L. B. Oliveira, "Siot: Securing the internet of things through distributed system analysis," in *IPSN*. New York, NY, USA: ACM, 2015, pp. 310–321.

[40] P. Cousot, R. Cousot, and F. Logozzo, "A parametric segmentation functor for fully automatic and scalable array content analysis," in *POPL*. New York, NY, USA: ACM, 2011, pp. 105–118.

[41] H. Nazaré, I. Maffra, W. Santos, L. Barbosa, L. Gonnord, and F. M. Quintão Pereira, "Validation of memory accesses through symbolic analyses," in *OOPSLA*. New York, NY, USA: ACM, 2014.

[42] V. Paisante, M. Maalej, L. Barbosa, L. Gonnord, and F. M. Quintão Pereira, "Symbolic range analysis of pointers," in *CGO*. New York, NY, USA: ACM, 2016, pp. 171–181.

[43] R. Poovendran, "Cyber-physical systems: Close encounters between two parallel worlds [point of view]," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1363–1366, Aug 2010.

[44] J. P. Conti, "The internet of things," *Communications Engineer*, vol. 4, no. 6, pp. 20–25, 2006.

[45] U. D. of Transportation, "IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environments WAVE," 2013.

[46] N. Patel, "90% of startups fail: Here is what you need to know about the 10%," 2015. [Online]. Available: https://www.forbes.com/sites/neilpatel/2015/01/16/90-of-startups-will-fail-heres-what-you-need-to-know-about-the-10/

[47] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[48] V. S. Miller, "Use of elliptic curves in cryptography," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 218. Springer, 1985, pp. 417–426.

[49] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé, "A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic," in *EUROCRYPT 2014*. Springer, 2014, pp. 1–16.

[50] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, 2006.

[51] NIST, "FIPS PUB 186: Digital signature standard (DSS)," 2013.

[52] ——, "Sp 800-56A recommendation for pair-wise key establishment schemes using discrete logarithm cryptography," 2006.

[53] D. R. Simon, "On the power of quantum computation," in *Symposium on Foundations of Computer Science (SFCS 94)*. Washington, DC, USA: IEEE Computer Society, 1994, pp. 116–123.

[54] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of ACM STOC 1996*. New York, NY, USA: ACM, 1996, pp. 212–219.

[55] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[56] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Deep Space Network*, vol. 44, pp. 114–116, 1978.

[57] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of ACM STOC '05*, ser. STOC '05. New York, NY, USA: ACM, 2005, pp. 84–93.

[58] C. H. Bennett and G. Brassard, "Quantum Cryptography: Public Key Distribution and Coin Tossing," in *Proceedings of IEEE ICCSSP'84*. New York: IEEE Press, 1984, pp. 175–179.

[59] J. Buchmann, E. Dahmen, and A. Hülsing, "Xmss - a practical forward secure signature scheme based on minimal security assumptions," in *PQCrypto*, B.-Y. Yang, Ed. Springer, 2011, pp. 117–129.

[60] D. A. McGrew, M. Curcio, and S. Fluhrer, "Hash-Based Signatures," IETF, Internet-Draft, 2017. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-sigs-06

[61] R. C. Merkle, "Secrecy, authentication and public key systems / a certified digital signature," Ph.D. dissertation, Stanford, 1979.

[62] NIST, "Post-quantum crypto project," 2016. [Online]. Available: http://csrc.nist.gov/groups/ST/post-quantum-crypto/

[63] D. McGrew, P. Kampanakis, S. Fluhrer, S.-L. Gazdag, D. Butin, and J. Buchmann, "State management for hash based signatures," *IACR Cryptology ePrint Archive*, vol. 2016/357, 2016.

[64] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, *SPHINCS: Practical Stateless Hash-Based Signatures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 368–397.

[65] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 3)," *NIST special publication*, vol. 800, no. 57, pp. 1–147, 2012.

[66] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[67] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *ASIACRYPT*, ser. LNCS, vol. 2894. Springer, 2003, pp. 452–473.

[68] A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” *IACR Cryptology ePrint Archive*, vol. 2012, p. 52, 2012.

[69] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in *Public Key Cryptography*, ser. LNCS, vol. 6571. Springer, 2011, pp. 53–70.

[70] Z. Liu and D. S. Wong, “Practical attribute-based encryption: Traitor tracing, revocation and large universe,” *Comput. J.*, vol. 59, no. 7, pp. 983–1004, 2016.

[71] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab, “Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks,” *Computer Communications*, vol. 34, no. 3, pp. 485–493, 2011.

[72] T. Kim and R. Barbulescu, “Extended tower number field sieve: A new complexity for the medium prime case,” in *CRYPTO (1)*, ser. LNCS, vol. 9814. Springer, 2016, pp. 543–571.

[73] A. L. M. Neto, A. L. F. Souza, Í. S. Cunha, M. Nogueira, I. O. Nunes, L. Cotta, N. Gentille, A. A. F. Loureiro, D. F. Aranha, H. K. Patil, and L. B. Oliveira, “Aot: Authentication and access control for the entire iot device life-cycle,” in *SenSys*. ACM, 2016, pp. 1–15.

[74] N. Mouha, “The design space of lightweight cryptography,” *IACR Cryptology ePrint Archive*, vol. 2015, p. 303, 2015.

[75] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, ser. Information Security and Cryptography. Springer, 2002.

[76] V. Grosso, G. Leurent, F. Standaert, and K. Varici, “Ls-designs: Bitslice encryption for efficient masked software implementations,” in *FSE*, ser. LNCS, vol. 8540. Springer, 2014, pp. 18–37.

[77] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov, “Design strategies for ARX with provable bounds: Sparx and LAX,” in *ASIACRYPT (1)*, ser. LNCS, vol. 10031, 2016, pp. 484–513.

[78] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalçin, “Block ciphers - focus on the linear layer (feat. PRIDE),” in *CRYPTO (1)*, ser. LNCS, vol. 8616. Springer, 2014, pp. 57–76.

[79] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, “The SKINNY family of block ciphers and its low-latency variant MANTIS,” in *CRYPTO (2)*, ser. LNCS, vol. 9815. Springer, 2016, pp. 123–153.

[80] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, “PRESENT: an ultra-lightweight block cipher,” in *CHES*, ser. LNCS, vol. 4727. Springer, 2007, pp. 450–466.

[81] J. Aumasson and D. J. Bernstein, “Siphash: A fast short-input PRF,” in *INDOCRYPT*, ser. LNCS, vol. 7668. Springer, 2012, pp. 489–508.

[82] S. Kölbl, M. M. Lauridsen, F. Mendel, and C. Rechberger, “Haraka v2 - efficient short-input hashing for post-quantum applications,” *IACR Trans. Symmetric Cryptol.*, vol. 2016, no. 2, pp. 1–29, 2016.

[83] J. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “BLAKE2: simpler, smaller, fast as MD5,” in *ACNS*, ser. LNCS, vol. 7954. Springer, 2013, pp. 119–135.

[84] M. Stevens, P. Karpman, and T. Peyrin, “Freestart collision for full SHA-1,” in *EUROCRYPT (1)*, ser. LNCS, vol. 9665. Springer, 2016, pp. 459–483.

[85] D. A. McGrew and J. Viega, “The security and performance of the galois/counter mode (GCM) of operation,” in *INDOCRYPT*, ser. LNCS, vol. 3348. Springer, 2004, pp. 343–355.

[86] N. Koblitz, “A family of jacobians suitable for discrete log cryptosystems,” in *CRYPTO*, ser. LNCS, vol. 403. Springer, 1988, pp. 94–99.

[87] D. J. Bernstein, “Curve25519: New diffie-hellman speed records,” in *Public Key Cryptography*, ser. LNCS, vol. 3958. Springer, 2006, pp. 207–228.

[88] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, “High-speed high-security signatures,” *J. Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[89] C. Costello and P. Longa, “Four‖: Four-dimensional decompositions on a ‖-curve over the mersenne prime,” in *ASIACRYPT (1)*, ser. LNCS, vol. 9452. Springer, 2015, pp. 214–235.

[90] S. Banik, A. Bogdanov, and F. Regazzoni, “Exploring energy efficiency of lightweight block ciphers,” in *SAC*, ser. LNCS, vol. 9566. Springer, 2015, pp. 178–194.

[91] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, “Triathlon of Lightweight Block Ciphers for the Internet of Things,” NIST Workshop on Lightweight Cryptography, 2015.

[92] P. C. Kocher, “Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems,” in *CRYPTO*, ser. LNCS, vol. 1109. Springer, 1996, pp. 104–113.

[93] B. Rodrigues, F. M. Q. Pereira, and D. F. Aranha, “Sparse representation of implicit flows with applications to side-channel detection,” in *Proceedings of the 25th International Conference on Compiler Construction, CC 2016, Barcelona, Spain, March 12-18, 2016*, A. Zaks and M. V. Hermenegildo, Eds. ACM, 2016, pp. 110–120.

[94] J. B. Almeida, M. Barbosa, G. Barthe, F. Dupressoir, and M. Emmi, “Verifying constant-time implementations,” in *USENIX Security Symposium*. USENIX Association, 2016, pp. 53–70.

[95] P. C. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *CRYPTO*, ser. LNCS, vol. 1666. Springer, 1999, pp. 388–397.

[96] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” in *CRYPTO*, ser. LNCS, vol. 1294. Springer, 1997, pp. 513–525.

[97] Y. Kim, R. Daly, J. Kim, C. Fallin, J. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, “Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors,” in *ISCA*. IEEE Computer Society, 2014, pp. 361–372.

[98] Y. Ishai, A. Sahai, and D. Wagner, “Private circuits: Securing hardware against probing attacks,” in *CRYPTO*, ser. LNCS, vol. 2729. Springer, 2003, pp. 463–481.

[99] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F. Standaert, “On the cost of lazy engineering for masked software implementations,” in *CARDIS*, ser. LNCS, vol. 8968. Springer, 2014, pp. 64–81.

[100] E. K. Wang, Y. Ye, X. Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow, “Security issues and challenges for cyber physical system,” in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int’l Conference on Int’l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 733–738.

[101] Y. Gu, Y. Yao, W. Liu, and J. Song, “We know where you are: Home location identification in location-based social networks,” in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–9.

[102] Y. Ge, B. Deng, Y. Sun, L. Tang, D. Sheng, Y. Zhao, G. Xie, and K. Salamatian, “A comprehensive investigation of user privacy leakage to android applications,” in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–6.

[103] W. Liu, E. K. Park, and S. S. Zhu, “e-health pst (privacy, security and trust) mobile networking infrastructure,” in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, 2014, pp. 1–6.

[104] A. Tekeoglu and A. S. Tosun, “Investigating security and privacy of a cloud-based wireless ip camera: Netcam,” in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, 2015, pp. 1–6.

[105] A. R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial internet of things,” in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[106] J. Hu, C. Lin, and X. Li, “Relationship privacy leakage in network traffics,” in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–9.

[107] J. Holvast, “History of privacy,” in *The Future of Identity in the Information Society*, ser. IFIP Advances in Information and Communication Technology, V. Matyáš, S. Fischer-Hübner, D. Cvrček, and P. Švenda, Eds. Springer Berlin Heidelberg, 2009, vol. 298, pp. 13–42.

[108] C. Gentry, “A fully homomorphic encryption scheme,” Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2009.

[109] F. Borges de Oliveira, *A Selective Review*. Cham: Springer International Publishing, 2017, pp. 25–36.

[110] R. A. Popa, N. Zeldovich, and H. Balakrishnan, “Guidelines for using the cryptdb system securely,” *IACR Cryptology ePrint Archive*, p. 979, 2015.

[111] F. Borges de Oliveira, *Selected Privacy-Preserving Protocols*. Cham: Springer International Publishing, 2017, pp. 61–100.

[112] F. Borges, P. Lara, and R. Portugal, “Parallel algorithms for modular multi-exponentiation,” *Applied Mathematics and Computation*, vol. 292, pp. 406–416, 2017.