Future Directions in Computing on Encrypted Data

Editor
Nigel P. Smart (University of Bristol),

Contributors
Dave Archer (Galois),
Dan Bogdanov (Cybernetica),
Sasha Boldyreva (Georgia Tech.),
Seny Kamara (Microsoft Research),
Florian Kerschbaum (SAP),
Yehuda Lindell (Dyadic Security and Bar-Ilan University),
Steve Lu (Stealth Technologies),
Jesper Buss Nielsen (Aarhus University),
Rafail Ostrovsky (Stealth Technologies and UCLA),
Jakob I. Pagter (Sepior),
Ahmad-Reza Sadeghi (T.U. Darmstadt),
Adrian Waller (Thales).

15. August 2015

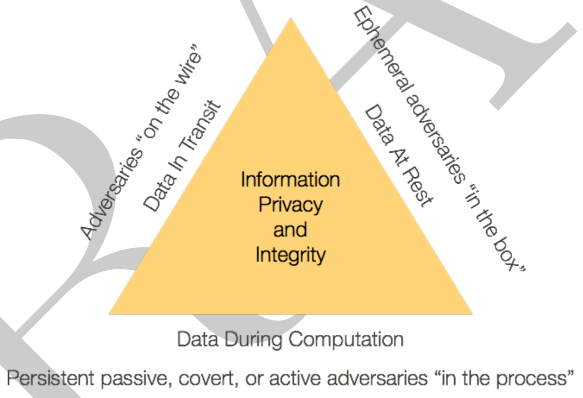
Revision 0.2

The work described in this report has in part been supported by the Commission of the European Communities through the H2020 ICT program under contract ICT-XXXXXX. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Contents

Executive Summary

Cryptography has long been used to protect data while at rest in persistent storage from adversaries who gain access to those storage media, and while in transit over network connections from adversaries that can monitor communications. As shown in the figure below, there is an important third pillar that cryptography provides to securing the life cycle of sensitive data where data may be at risk of privacy or integrity loss: namely during computation. The first two pillars have been deployed in multiple applications scenarios over the decades, with deployment increasing to almost ubiquotous levels over the last second. The third pillar, namely protecting data during computation, has only become practically possible in the last decade and we are starting to see the first commercial experimentation and deployments. It is to this third pillar of cryptography that we address this report.


Adversaries "on the wire"
Data In Transit
Information Privacy and Integrity
Ephemeral adversaries "in the box"
Data At Rest
Data During Computation
Persistent passive, covert, or active adversaries "in the process"

The report discusses three key technologies Searchable Encryption, Fully Homomorphic Encryption and Multi-Party Computation. We give an overview of what each technology can achieve, and what it currently cannot do. We then go on to discuss the known existing prototypes and products in this space, before examing various barriers to adoption which the technology faces. Finally we outline general open research and development problems which need to be solved if this third pillar is to become as ubiquotous as the first two pillars.

Our main conclusion is that techniques for searchable encryption are now very well developed and are beginning to find commercial application, as evidenced by the increasing number of start-ups/SMEs in this space. Techniques for general multi-party computation are now very well advanced and are beginning to find commercial deployment, there are fewer start-ups/SMEs in this area; most of which are based outside of the US. The commercial outlook for Fully Homomorphic Encryption is however, in our view, less well defined without considerably further basic research being performed.

Chapter 1

Introduction

This document summarizes the input from various experts around the world on the topic of Computing on Encrypted Data (COED). The benefits of encrypting data are well known, however as soon as data is encrypted it becomes seemingly impossible to perform computations on the data without decrypting it first. This need to perform decryption means that in practice

- Either the data is not encrypted in the first place,
- Or modern storage and processing facilities (such as cloud and/or distributed computing) cannot be applied.

Surprisingly, there are technologies to allow for computation to be performed in the encrypted domain. These technologies are becoming increasingly well developed, and some have found commercial application. In this report we outline the technologies, the current known deployments, and open problems and research directions which we believe are needed to be followed so as to capture the full potential of this technology.

Data sets are so large now that the idea that one keeps a local copy of data on ones disk and performs analysis on it is a thing of the past. Often, it is also the case that the data cannot be aggregated at a single repository due to mutual privacy concerns. Thus we are already seeing "Data-as-a-Service" (DaaS) applications arising. An obvious example of this is of search (e.g. Google), the numerous biological databases online, or government statistical data. This is likely to continue. Another driver is that data breach is now common, and attackers are within a corporate network. This leads to the need to keep even internal data under the protection of encryption; think HIPAA or PCI compliance for mandated controls in this respect. However, the owners of this data still want to compute on it, despite it being encrypted. Applications of new ways of doing business can also be opened up by COED technologies. For example the ability to conduct auctions in a secure manner opens up new ways of enabling markets to obtain the clearing price. Another (paradoxical) example is the ability for different corporate entities to compare if they are under a similar cyber-attack, without disclosing even the fact that they are under attack to each other unless and until both simultanesouly learn that they are suffering from the same attack. By doing this in a manner which preserves privacy of inputs one can obtain more economically efficient outcomes, and even outcomes that were not previously possible where competitive or legal restrictions prevent information sharing.

We start the report by outlining the three main technologies which support COED; these are technologies which support Encrypted Search such as Searchable Symmetric Encryption

(SSE), as well as more general technologies such as Fully Homomorphic Encryption (FHE) and Multi-Party Computation (MPC).

1.1 Encrypted Search

The most basic problem one has when encrypting data is to perform a search on it. For example, suppose one encrypts the NHS health records and places the encrypted records on a server. A Doctor may then want to retrieve all of a given patient's records. However, to do this they need to search for those records containing the patient's names. But if the name is encrypted this becomes seemingly impossible, since secure encryption is a probabilisitic process.

There are three forms of "encryption" which allow such searches to be carried out. We list these in increasing order of security, and explain each of them as they often cause confusion in non-technical communities.

- **Tokenization:** In a tokenization scheme certain records in a data base, for example someones name, are replaced by a random looking "token". For example one may replace the name by the cryptographic hash of the name along with some secret key. With such schemes the data is not encrypted it is obfuscated; since, given a token, one is unable to recover the original input data, however anyone with access to the input data query and the key can form a token which can then be searched for.
- **Property Preserving Encryption:** This includes various forms of encryption such as deterministic encryption and Order Preserving Encryption (OPE). A deterministic encryption scheme is one in which every time one encrypts a plaintext one obtains the same ciphertext. Deterministic encryption schemes act much like tokenization schemes; however one has the added functionality of given the token and the key one can recover the original plaintext. However, it has been known since the early 1980s that deterministic encryption offers very little security compared to randomized encryption algorithms in general algorithms. Order Preserving Encryption is a form of encryption which takes an ordered universe of plaintext, for example integers, and produces ciphertexts which can be compared; in the sense that a relation is defined on ciphertexts which mirrors the relation on the plaintexts. OPE schemes thereby allow sorting of plaintexts, by sorting the ciphertexts. In specific applications both can provide a suitable level of security, but for general applications deterministic encryption and OPE may not be a suitable technology for COED.
- **Searchable Encryption:** Searchable encryption (and more generally structured encryption) are encryption schemes that encrypt data structures in such a way that they can be privately queried. To search on encrypted data, one encrypts a search structure (e.g., an inverted index) and uses a standard encryption scheme (e.g., AES) to encrypt the documents. To query the encrypted documents, the client provides the server with a special token that encapsulates the keyword and that, when combined with the encrypted structure, lets the server find out which (encrypted) documents it should return. The structure is encrypted in a randomized way but the search operation reveals some information to the server. Such searchable encryption schemes, which only makes gen-

eral sense for symmetric key algorithms[1], provides a randomized encryption algorithm; i.e. each encryption of the same plaintext produces a different ciphertext. Encryption methods for SSE provide a form of indistinguishability of encryptions (IND), which is the de-facto security standard for encryption schemes.

A key issue with searchable encryption is what type of search is enabled: e.g. basic keyword search, or Boolean queries, or substring search, or range queries, or similarity queries, or ... Solutions vary greatly in terms of security, ease of use and efficiency they achieve, as, not surprisingly, these properties are at odds with each other. For example, symmetric deterministic encryption yields a very efficient and simple solution for basic keyword search. In addition, no changes are required on the server side, and updates are not an issue. Similarly, Order Preserving Encryption (OPE) gives an efficient solution for range queries, that again does not require server side changes. However, security of these solutions is not very strong. Better security can be achieved with encrypted index based solutions, such as in full blown SSE. But most of these require server changes and the index structure becomes complicated.

A problem with all such schemes listed above (in their pure form) is that if the searching is performed on a remote server, then an eavesdropper, or even the server, may be able to recover information from the access patterns, or other associated metadata, revealed by various queries. For example in our health care example, all the records returned to the doctor can be reasonably deduced to be patients of the said doctor. Cross correlation between various queries, or indeed various databases, can be used to reveal identities of peoples private data[2]. Thus on its own searchable encryption does not provide full security, and often needs to be combined with techniques such as Oblivious RAM (ORAM).

Oblivious RAM, PIR and Garbled RAM is a technique to obtain queries from a memory, and update the said memory, without revealing the memory locations which are being read and written. Using usch technology prevents the data access patterns from being revealed. For example, when an entity that holds encrypted data it could figure if if some particular (encrypted) data has been used multiple times across different queires. This *metadata* information is also possible to protect by using a host of more advanced techniques, both public-key based and private-key based, based on ORAM style techniques. The performance of these algorithms differs depending on database size and specific setting.

1.2 Fully Homomorphic Encryption

A Fully Homomorphic Encryption (FHE) scheme is a randomized public key encryption scheme which allows arbitrary computations to be performed on encrypted messages. Suppose we have a function $F(x, y)$ on two data items x and y, then given an encryption c_x of x and an encryption c_y of y an FHE scheme allows one (in theory) to compute a ciphertext $c_{F(x,y)}$ which encrypts $F(x, y)$ given only the input of c_x and c_y. Whilst theoretically any such function can be computed, in practice one is limited to relatively simple functions with current technology. For example it is relatively straight forward to use FHE technology to compute the encryption of a mean or standard deviation of a set of encrypted data.

[1]Public Key Encryption with Keyword Search (PEKS) schemes are possible, but any scheme in which the search token can be publicly generated are susceptible to exhaustive keyword search.

[2]The most famous example of this cross correlation being the de-anonymization applied to the Netflix dataset, which did not even use access patterns to perform the de-anonymization. See `https://www.cs.utexas.edu/~shmat/shmat_oak08netflix.pdf`.

A key aspect of FHE technology is the use case: Anyone can encrypt data (it is a public key scheme), anyone can perform computations on the encrypted data (for example a remote cloud server), but only the designated secret key holder can decrypt. This implies that FHE is more suited to applications in which one wishes to outsource computation to a remote server. If multiple decryption key holders are required in an application then all schemes can support threshold decryption, or decryption with respect to any access structure.

If computation is performed by a remote server, then the decryptor may want some guarantee that the data he decrypts is actually the correctly computed function. To protect against such active attacks by the remote computation server other technologies can be applied, for example Verfiable Computation.

Fully Homomorphic Encryption if ever fully practical would enable applications such as enabling a user to hold encrypted email at Google, and still allow Google to perform arbitrary search queries on the encrypted emails, returning the query results to the user in encrypted form for the user to decrypt. However, with current technology such an application is a very long way off since current scheme are very inefficient when processing even small functions on encrypted data. However, the eventual promise of such a technology has resulted in funding agencies around the world placing a lot of investment in the whole COED area in the last few years.

A slightly weaker primitive then Fully Hommomorphic Encryption is encryption that allows either only additions and no multipications or only a limited number of additions. Such schemes, unlike FHE are fairly efficient, thus applications where a task can be descrbied by a function that required very few (nested) multiplications can be performed relatively fficiently. Despite inherent limitations this class is rather large and includes, for example, private keyword search and some forms of electronic voting.

1.3 Multi-Party Computation

Whilst FHE and SHE allow computation to be performed on encrypted data held on a single server, MPC allows computation on data which is "split" between multiple servers. As a simple example consider some numerical data x and suppose we randomly split x into two numbers x_1 and x_2 such that $x = x_1 + x_2$. Then we can place x_1 on one server and x_2 on another server. This is a form of encryption, as we can think of the data x_1 as a key, and the data x_2 as the encryption of x under the key x_2 (or vice-versa). Such a splitting of data is called a *secret sharing*, and MPC allows one to perform computation on such secret shared data. Secret sharing can be performed between any number of servers, and with any desired access pattern.

Technology to perform MPC in theory has been around since the mid 1980s, but it is only in recent years that progress has been made to make MPC a practical reality. There are a number of different methodologies to perform MPC, some based on a technique called a Garbled Circuit due to Yao, and some based directly on the secret sharing ideas mentioned above. Just as with FHE above, one needs to protect against a server/party deviating from the protocol. Almost all modern MPC protocols provide such protection. The additional concern in MPC is whether several participants collude to jointly deviate from the protocol and have a joint strategy; techniques that prevent even such attacks are known to exist. In its basic form MPC is exists in two regimes: honest majority and dishonest majority (in particular, two-party computation) – the techqnieus are somewhat different in each regime.

At a high level MPC allows a set of n parties to contribute their own input data $x_1, \ldots, x_n$ and then to jointly compute the output of any function on the input data, $F(x_1, \ldots, x_n)$. The classic “toy example” being the problem of a set of bankers deciding who will pay for dinner on the basis of who got the biggest bonus. They can each enter their own bonus value x_i, and then the function will output the player number with the largest bonus.

Practical systems for various forms of MPC have been built and are deployed. At present relatively large functions can be computed relatively efficiently; however the bottleneck in MPC protocols is that computation of complex functions requires large data exchange (either in terms of the total bandwidth or the number of rounds of communication, depending on the protocol). This should be compared with FHE based systems, which give very low communication complexity, but are impractical due to the large amount of computation required.

Chapter 2

Potential Markets Not Yet Realized

Today, it is clear that consumers have embraced cloud services. Most users are turning to personal cloud storage services like Dropbox, OneDrive or Facebook to store and backup their documents, music and photos. Similarly, many users today use web mail, effectively outsourcing their email to a cloud provider. Much of this data is personal and private and, given the prevalence of data breaches from Government, healthcare and corporate systems, there is a possibility that consumers and/or regulators will prefer services that provide strong security and privacy over ones that do not. This potential increase in the demand of privacy-enhanced services is motivating venture capitalists and government agencies to invest in advanced cryptographic technologies and, in particular, COED technologies.

Ideally, COED complements cryptographic protocols for protecting data in transit and ciphers for protecting data at rest, closing the vulnerability gap for sensitive information by allowing data to stay encrypted at all times (after initial ingestion). Even a simple form of computation on encrypted data, as enabled by for example by SSE, would be an immediately applicable and, in fact necessary, component of any end-to-end encrypted service. It could enhance end-to-end encrypted cloud storage or email (including webmail, hosted mail or standard mail) with search capabilities. It could also be used in a non-end-to-end setting to add search functionality to the encrypted backend systems of cloud providers.

Enterprise customers have also embraced the cloud but so far seem reluctant to use it for sensitive or mission-critical data. Some of their concerns include potential data breaches at the cloud provider, insider attacks, and questions about liability and regulatory compliance. Providing searchable end-to-end-encrypted enterprise-level services would go a long way to addressing those concerns and, indeed, several startups are already doing this. But enterprise customers could benefit from COED even in on premise scenarios. For example, the ability to process encrypted data would mean they could keep a lot of their data encrypted at all times, greatly reducing the risk of data breaches and insider attacks. For example, consider the Home Depot breach, where an APT buried in point-of-sale terminals exploited a brief moment in transaction processing where data was not encrypted in order to exfiltrate personal information. Encrypting all the data at rest, and *never decrypting* it during the processing will enable enhanced security, and help mitigate against such breaches.

For customers to outsource their most sensitive data, they must have a high-level of trust in their cloud provider. While it is certainly true that large cloud providers can secure data better than most businesses there are still many reasons why a cloud provider might want to offer searchable end-to-end encrypted services. In some cases, customers cannot outsource certain datasets due to regulation. In other cases, a customer might be concerned about government surveillance or espionage. Or perhaps a the cloud provider itself might want to limit its own liability, by preventing *rogue* employees from releasing information

By using, for example, MPC cloud providers can split their operations across multiple geographic locations. Thus mitigating risks from local law enforcement agencies to request keys to decrypt a specific record (since all keys from all geolocations would be needed to perform this task). Of course, such guarantees have become even more important since the Snowden disclosures and this has become an important concern for large US cloud providers.

Since the Snowden disclosures, we have learned of the massive surveillance infrastructure developed by the NSA, GCHQ and other intelligence agencies. While one can debate the value of this infrastructure for national security what is clear is that the data it generates and stores is profoundly intrusive and sensitive. As such, since 2014, the United States Government has been debating how to reconcile this infrastructure with the privacy afforded to US citizens by the 4th Amendment of the US constitution[1]. Of particular interest is the NSA bulk metadata program which stores and analyzes the phone call metadata of US citizens. SSE and structured encryption, combined with other technologies like MPC, could be used to design privacy-preserving alternatives of the surveillance infrastructure, in effect obviating the false choice between national security and privacy.

Breaches of government data are not unheard of, recently the US government OPM breach exposed significant unencrypted personal information of millions of holders of US security clearances, arguably the worst infosec failure in US history. Symantec[2] reported 312 major breaches of unencrypted data in 2014, exposing 348 million identities. Particularly worrisome is that the largest proportion of those incidents (116 incidents) was in the healthcare sector, perhaps unsurprising because the underground market valuation for healthcare information ($10 to $50 per identity) has far surpassed that of credit card information ($0.50 per identity). COED could substantially reduce the impact of such breaches by providing the means to perform some computations while data remains encrypted, potentially removing the need to leave data "in the clear" at any point in its lifecycle.

Although of the three fundamental technologies, FHE receives most attention in the technical media, it is perhaps the least practical. In particular, it is slow, decisions and comparisons are difficult to do, and it is only really suited to outsourced (and relatively elementary) computations, which limits its applicability in real-world scenarios. On the other hand SSE is becoming more practical and efficient, but again is mainly limited to outsourced data storage, though more advanced computation is coming into play. Put another way FHE allows outsourced, but non-collaborative, computation and SSE enables secure storing but not sharing of data.

Although useful, these are not the main business pulls for privacy and confidentiality protection of data. More significant privacy and confidentiality issues for businesses and governments lie in collaboration and sharing of data. In this respect MPC is the most promising technology from this perspective as it is focussed on secure collaboration, including verifiability of results.

Another area ripe for the promise of COED is in the sharing of threat intelligence between companies. Symantec notes that such sharing is essential to security, but that companies fear

[1] *The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures.*

[2] Symantec's 2015 Internet Threat report is available here `https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932_GA-internet-security-threat-report-volume-20-2015-social_v2.pdf`.

sharing such information with other companies. Cyber attacks are increasingly distributed and global in scale, and responding to and protecting oneself from attacks requires increased sharing of intelligence and other data. At the same time, sharing of network data can be difficult or impossible to do because of privacy and confidentiality concerns, as well as the fear of opening oneself up to negative publicity. MPC could allow parties to collaborate in detecting attacks and gathering intelligence while maintaining privacy and confidentiality. It also makes possible new models of service: As a simple example, Company A may specialize in providing attack detection algorithms and wish to maintain their IP by keeping these secret. On the other hand, Company B may be interested in attack detection services on their networks, but be unwilling to provide network data to Company A due to confidentiality concerns. MPC solves this problem, and opens up this market to smaller and less “trusted” attack detection services.

Many of these potential COED application areas require capabilities that today’s COED technology does not offer: efficient, correct, and secure applications that can be created, verified, maintained, and deployed by non-cryptographers.

Chapter 3

Current Prototypes and Products

There has been considerable progress in research prototypes for specific problem instances in recent years; and in addition a number of companies are now fielding products in the field. We outline a number of these below.

3.1 Known Prototypes Built

The DARPA funded PROCEED programme resulted in a number of research prototypes based on MPC for various use cases, these included: Satellite Conjunction (Stealth (2PC) and Cybernetica (3PC)), merging encrypted VOIP streams (Galois (3PC)), encrypted email filtering (Galois (3PC)), Bayesian spam filtering (Cybernetica (3PC)), secure route mapping (Georgia Tech.), fair Resource Allocation (Stealth (2PC)). The IARPA SPAR programme resulted in prototypes of large-scale SSE with additional functionalities, such as boolean queries, approximate search and wildcards, range and keywords.

In Europe a number of EU and national funded projects have resulted in prototypes including: secure surveys (Cybernetica), financial benchmarking (Alexandra Institute, Partisia and Cybernetica), collaborative genome analysis (Cybernetica), secure Supply Chain Management (SAP), human face recognition (Various research groups).

In the case of secure database access a number of prototypes have been built, including ones based on SSE and ORAM technologies (Stealth), ones based on MPC (Cybernetica), and two systems based on various technologies called CryptDB and Mylar (MIT). The CryptDB and Mylar systems have been used as a building block in other prototypes by Google (Encrypted BigQuery), Lincoln Laboratories, SAP, and a hospital in Boston. Microsoft Research have built a prototype which enables a searchable encryption functionality within a Microsoft Exchange environment.

3.2 Known Products

In the field of products that enable search on encrypted data there are a number of companies marketing solutions in this space: Aikicrypt, Bitglass, CipherCloud, CryptonorDB, Krypnostic, Oblivilock, Private Machines Inc, Skyhigh Networks, Vaultive Security, and ZeroDB. The underlying technology used by each company is often, due to commercial considerations, hard to fathom from the outside (for example are they using deterministic or searchable encryption, or just tokenization?).

A few companies provide products based on Multi-Party Computation technologies. Interestingly all are based in Europe and Israel. These include Cybernetica which market a secure database accessed by MPC technology as well as other applications (including systems for detecting tax evasion in Estonia); Partisia which markets a system for secure auctions; Sepior which markets a system to provide secure key management in the cloud; and Dyadic Security which markets systems to reduce the risk of server compromise by splitting sensitive data via secret sharing.

Chapter 4

Barriers To Adoption

After decades of theoretical research, COED is now a mature technology. Due mainly to major algorithmic improvements, there are real problems that can be solved today using COED. In addition, industry has heard about secure computation and is interested in utilizing it for business purposes. One of the problems that academia faced in the past with respect to COED technologies is a lack of focus regarding the problems that the "world" is interested in solving. Since industry has now heard of this technology, and wishes to use it, we now have an opportunity to learn about brand new problems where secure computation can be applied. The ability to solve these problems satisfactorily will require additional academic research which will in turn generate further industry interest. We therefore envision a healthy cycle of academic/industry collaboration around the design of secure computation solutions and their applications in practice. However, despite this significant barriers to adoption remain. We outline a few here.

4.1 Business Risk

Several factors lead to the risk of investment in COED being hard to judge. There are many alternative approaches, with options and choices for each approach, making initial selection difficult. Tailoring and optimisation is often needed for particular applications to achieve suitable performance. This is currently done on a case-by-case basis making it costly, and hard to judge in advance the cost and risk of using COED. In addition, there is a large gap between application software and algorithms and low-level COED primitives, which also increases the cost and risk.

4.2 Useability

In a recent international group study to understand the barriers of adopting secure computing in the real world[1] it was found that the potential users expect usability as-good-as or better-than that available with normal tools. For example users used to performing statistical queries would expect an interface similar (if not the same) to that of R, or those performing database queries would expect interfaces the same as those in SQL.

[1]See `http://www.usable-security.eu/`, and in particular `http://www.usable-security.eu/workpackages-and-reports/wp1-requirements-gathering/d12-requirements-specification-based-on-the-interviews`.

4.3 Business and User Education

There is a need for bootstrapping in terms of getting the technology accepted to solve security issues, e.g. in relation to a specific probelm. Often, such a process is more smooth if there are concrete examples to point to. Hence there is still a need to develop research prototypes and demonstrators to provide items to point to.

There is a need to clearly communicate the kinds of problems that these technologies can solve. From a purely technical perspective, the landscape is very heterogeneous . There is a challenge for the community to clearly communicate to funding instruments and decisions makers what the "practical" potential of the technology is. In theory COED technologies can be said "to solve all your problems", as almost any security task can be framed as an MPC problem; but in practice this is not the case. Practical systems need to integrate with existing frameworks, human users, and legal and societal aspects. These technologies have so much potential related to many of the current main technological trends, but we need to be careful in how we present this.

4.4 Standardization

Standardization can be achieved on multiple levels. At the level of API standardization there has been some attempts in multiple research projects, but to little success. Currently, compatibility has been achieved on a service level, e.g., PRACTICE demonstrated an Enterprise Java-based survey system that works on both Cyberneticas Sharemind and Partisias Fresco platforms. Many challenges still remain, e.g. having multiple runtimes support the same programming interfaces and so on.

On the cryptographic level, standardization of MPC and FHE is taking place in ISO/IEC JTC 1 SC 27 where the ISO/IEC 19592 (parts 1 and 2) project is standardizing secret sharing (editors Dan Bogdanov, Shinichiro Matsuo and Koutarou Suzuki) and ISO/IEC 18033 (part 6) will standardize homomorphic encryption (editors Pascal Paillier and Atsuko Miyaji).

There are also higher-level standards, e.g. ISO/IEC 29101 (Privacy Architecture Framework) describes secure computing as an optional component. There are also plans in SC 27 to include these technologies in upcoming privacy standards.

4.5 Legal and Regulatory Issues

Even if data is encrypted the legal and regulatory framework may not allow it to be placed on a cloud server, or indeed any third party server, without the data subjects express permission[2]. This on its own may limit the eventual usage of COED technologies. However, it is unclear what encrypted means from a legal perspective. For example, is secret shared data encrypted? From one perspective one can think of each share as a key, and hence data independent, yet from another perspective we can think of each share as an encryption of the data under some key derived from the other parties shares.

Even if data encryption is allowed, one needs to secure the key to the data. Here COED technologies can provide additional protection by enabling a key to be split into parts and

[2]The EU funded PRACTICE project has done some preliminary work in the EU context of legal issues in this area, see `http://practice-project.eu/downloads/publications/D31.1-Risk-assessment-legal-status-PU-M12.pdf`.

distributed in different geographic regions. But again legal and regulatory compliance might require government access to said keys, which may in some applications defeat the increased security obtained by geographic distribution of key shares.

Thus we see a tension in that the legal and regulatory framework is not keeping up with technology developments. Yet governments are driving to enable more data sharing and processing of citizens private data. Yet this could be done in a way which engenders a larger amount of trust if COED technologies were deployed, yet without clear legal and regulatory guidance there is a conflict between the right of a citizen to data privacy, and the benefit to society of large scale processing of data (for example to enable personalized medical care).

4.6 Competing Technologies

There are always many ways to skin a cat, and the COED technologies of SSE, FHE and MPC (and their variants) are just one such way. In the original paper on FHE in 1978 by Rivest et al. a second solution to the problem on computing data was presented; namely an encrypted processor. In the encrypted processor data would be held in RAM or on disk in encrypted form; say under an AES encryption key. Then when the data is loaded into the processor it is decrypted and placed in the processors registers. Computation within the processor is then performed on clear data. Once a write back occurs to memory the register data is encrypted once again. In 1978 this was dismissed due to high costs of encrypting and decrypting the memory bus, and the relatively costly Instruction Set Architectures of the time.

Modern processors however work on a RISC principle in which memory access instructions and data processing instructions are considered to be distinct. Thus such an architecture essentially involves only a minor modification of the memory access instructions on a RISC machine, and with current AES hardware it is not so costly compared to the time needed to access RAM in the first place.

This philosophy has been adopted by Intel in its new SGX instruction extensions. Such a philosophy will allow computation on data which is held encrypted in memory, or on disk. Whilst this may solve a number of the issues related to storing data in the clear we do not see it as a technology which can replace all possible applications of COED technologies. Indeed, as we outline below, there are a number of cases in which SGX technology can be combined with COED technologies.

4.7 Compliance

Secure computation looks like a match for financial institutions, especially when it comes to hazard modelling, risk analysis, fraud detection and similar applications. We are seeing interest in this sector, but the customers are looking for easy-to-integrate tools, familiar interfaces and mature, tested implementations. All this needs further development of prototypes and demonstrations showing that the technology has gone beyond theory.

However, much of the financial institutions motivation for adopting a specific technology is based on compliance and auditability. Thus new disruptive technologies are locked out by compliance regimes which specify the required use of legacy systems. For example in banking HSMs are required for compliance reasons, and any new disruptive technology to HSMs (which COED technologies could provide) are essentially locked out. A related element is the ability to clearly explain why a COED approach is better than alternative legacy approaches.

A second form of compliance which prevents technological adoption, is the use of controls and legal frameworks to ensure data is processed in a “secure” way. Here the legal framework may dictate who can access data etc., but does not insist that this is technologically enforced, it is simply the risk of legal action which is meant to compel the organization to adopt good practice. Hence, a superior technical solution is not adopted due to an entrenched belief in legal and regulatory frameworks being enough.

Another threat to the adoption of COED technologies is that alternative approaches relying on trusted third parties, or simply accepting the risk of sharing data, has become entrenched, as these will remain a less costly and risky approach from a purely monetary business perspective. In such a scenario security is maintained by human oversight, human controls and audit logs; namely the instruction to a staff member of not to look into some dataset is deemed sufficient to ensure that they do not. The threat of disciplinary action if caught is assumed to be a sufficient deterrent.

However, this reliance on such “soft” frameworks such as legal enforcement, and/or human oversight, is beginning to be brought into question by the increased sophistication of data breaches. Whilst compliance based solutions might work for the systems and people you trust, often large organizations and government agencies are unable to trust their own systems and people due to their size. In such a case a technical solution such as COED which removes the need to place trust even in your own systems and your own employees will make increased business sense.

Chapter 5

Open Research Problems and Directions

We first touch on technology specific open research problems and directions, and then we cover a much larger range of problems which cross the entire technology landscape in this area. It is interesting that most of the problems are at the higher level of use, deployment and integration as opposed to at the fundamental level of science. This shows an increased maturity of the field in that a lot of the basic “scientific” research has now been completed, and we are now moving into a new more “engineering” phase of the development of COED technologies.

5.1 SSE

The existing SSE solutions vary greatly in terms of security; but understanding security of each construction, and comparing security of various schemes is not easy even for cryptographers. Security statements for many SSE schemes specify a leakage profile (as some information needs to be leaked to enable efficient search). But these leakage functions are often very complicated and unintuitive. We shall return to the question of leakage below, as it applies to a number of COED technologies.

The other challenge is to have more solutions covering wider class of queries. The existing SQL-type queries for unencrypted structured data can be a guide. For each type of query, it is good to have various solutions that provide flexibility in terms of efficiency, security, easy of update, server-side changes requirements, etc. SQL offers a very rich set of features that is currently not supported by SSE:

- Aggregations can only be performed using homomorphic encryption which then does not allow any further processing.
- Range queries can be efficiently supported by order-preserving encryption, but security is quite weak. There are other solutions with much better security possible, but they either require a lot of interaction or very inefficient.

The other challenge is to have more solutions covering wider class of queries. The existing SQL-type queries for unencrypted data can be a guide. For each type of query, it is good to have various solutions that provide flexibility in terms of efficiency, security, easy of update, server-side changes requirements, etc SQL offers a very rich set of features that is currently not supported by much of the open literature on SSE:

- Aggregations can only be performed using homomorphic encryption which then does not allow any further processing.
- Range queries will fundamentally supported are very inefficient.
- Substring queries are either insecure due to storing essentially deterministically encrypted characters or very storage-inefficient, since they require to store all substrings.
- Updating index-based SSE is still not fully solved. We need to investigate methods for efficiently updating the index of SSE.

We need to investigate how to extend the basic model of (index-based) SSE to more search functionality in order to meet the requirements of developers. This can be in the form of new application libraries that transform the queries or in the form of new SSE schemes.

5.2 FHE

Fully Homomorphic Encryption is currently impractical. What is practical is something called Somewhat Homomorphic Encryption (SHE) which allows for evaluation of circuits of low multiplicative depth. Using packing techniques it is relatively efficient to evaluate a low multiplicative depth arithmetic circuit on multiple data items at once. Thus with present knowledge one can only see SHE solutions being used in very limited application areas, or as a sub-component within another system. For examplet the SPDZ MPC system uses SHE as part of its pre-processing phase.

Without significant foundational research we do not see that FHE will become practical in reality, and SHE will continue to be of limited use in niche applications. Due to these facts we do not consider them much in this report which focuses on more near market technologies for computing on encrypted data.

5.3 MPC

We draw an analogy between MPC today and general-purpose programming in the mid-1950s. The year 1952 saw the first compiler, targeting Glennies autocode for the Manchester Mark 1. Like many of todays MPC languages, this autocode was highly specific to underlying infrastructure, allowing essentially no abstraction or reasoning. Brookers 1955 autocode for the Mark 1 was relatively machine-independent, but its compiler provided no optimization – not so different from current compilers for MPC languages. In 1955 we saw FLOW-MATIC for UNIVAC-1, motivated by business needs for easily understandable programming languages. Fortran, also motivated by this need in other domains, appeared in 1957. Similarly today, the Python-based SPDZ language (Uni. Bristol), SecreC language running on Sharemind (Cybernetica), Wysteria (Uni. Maryland), and ObliVM (Uni.s Cornell and Maryland) have made progress toward easy-to-use languages for MPC. Significant optimization and instrumentation facilities only appeared for general programming in the late 1960s and early 70s, and verified compilers appeared much later. In a similar way, relatively little work has been published to date on optimizing or verified compilers for MPC. We feel that MPC is at a point where automatic optimization for performance and security is an achievable next step toward making secure computation adoptable by the general programming community.

Significant performance improvements are still needed, especially when large data sets are to be considered. The performance of protocols for secure computation can be boosted by running them on bespoke hardware, or outsourcing a component of the computation to such hardware. There are a number of possible directions here:

- The use of MPC-systems on chip. These have already been proposed and investigated as a means of preventing side-channel attacks on smart cards; in which case the technology is usually referred to as “masking”. But such techniques could also be made to create more secure trusted hardware modules.
- The use of trusted hardware add-ons to provide increased performance. For example in triple generation for Beaver style MPC protocols, such as SPDZ and Tiny-OT, or for producing garbled circuits in Yao or Garbled RAM based protocols.

The current state of affairs is that we have protocols that have very low communication complexity but high round complexity (these are the GMW-based protocols like Tiny-OT, SPDZ, and so on) versus protocols that have high communication complexity but low round complexity (these are the Yao-based protocols). In many cases, communication is the bottleneck, and neither of these alternatives is satisfactory. Specifically, low bandwidth is not very helpful if it requires high-round complexity between parties that may be physically far from each other. Likewise, Yao-based protocols require sending large Garbled Circuits which become the bottleneck of the computation. An important challenge is to design protocols that have both low bandwidth and a small number of rounds.

On the other hand some communication allows one to produce more efficient representations of the functionalities that users wish to compute. For example the ability to “open” masked data, in secret sharing based systems, allows efficient computation on data items such as floating point numbers. Hence, methodologies need to be developed which use the minimum amount of communication needed to express the function efficiently, but no more, would be of great benefit.

The tradeoff between efficiency and generality is a difficult one in any area. When privacy and adversarial behavior are added into the mix, the complexity becomes all the more challenging. Existing tradeoff approaches to general secure program computation typically fall into a few categories, and within each category, several tradeoffs can be made: including the important tradeoff of round complexity (multi-round vs. constant round). One must analyze the feasibility of existing approaches as well as bring new ideas to the table on how to improve the practical efficiency of solutions in this area. The goal is to compare and contrast existing schemes, new pathways, and even new tradeoffs in old schemes.

5.4 General

In this section we outline a number of research directions which are relatively general and apply to a number of COED technologies.

5.5 System Issues For Storing Encrypted Data

When storing encrypted data an important customer requirement is to frequently change the encryption key. Downloading the data, decrypting, encrypting under the new key and uploading can be a considerable cost in FHE and SSE technologies. Even in MPC based solutions there can be issues in re-encryption depending on the underlying MPC technology being used. Proxy re-encryption is a promising alternative for FHE and SSE solutions that needs to be fully investigated, as it will allow this operation to be performed on the server. Ideally, the proxy re-encryption will be both uni-directional and multi-use.

A cloud service usually never stores the data of one client only, but of multiple clients. Multitenancy models allow the efficient sharing of resources among clients, but still provide the necessary separation of control. In order to ensure this separation different encryption keys can be used (in the case of FHE and SSE), or different sharing structures (in the case of MPC). However, there is always the use case of data sharing and hence we need to search over data encrypted under different keys.

5.5.1 Integration of Technologies

One technical challenge for technology transfer is integration. All existing COED technologies tend to build monolithic systems of the form "give me a function f describing the problem you want to solve securely and I will give you a protocol doing this for you for this particular trust structure on the participants". This is, however, not how large commercial software is built. Large software systems are built by evolving existing solutions, integrating enterprise applications, gluing together parts developed by different companies and technologies. In distributed systems similar but often-easier problems have given rise to a multitude of network formats, message brokering systems and enterprise application integration systems for such integration. COED research seems far behind on this aspect. Can we fit into existing integration methodologies or do we need completely different solutions. We believe that the area can be a basis for interesting fundamental research and technology development. We see three different aspects, as described now

- **Intra COED Technology** We need an increased research focus on how to integrate different COED technologies. For instance, how do we best (most efficiently/most securely) move data from a secret sharing based MPC technology into a FHE technology, or to a SSE based technology, and vice-versa? Can we develop something like a network format which it is easy to convert to and from for most COED technologies? How do we best deal with the fact that the trust structure might be different? There seems to be plenty of room for fundamental research on developing solutions and investigating inherent limitations.

- **Other Security Technology** COED will have to integrate with other solutions to the same problem. As with most other technological breakthroughs in the past, COED will not replace other (possibly worse) solutions to the same problem, but will have to live along with them and integrate with them and in the best case very slowly replace them. We therefore need ways to integrate with other security solutions, like security based on secure hardware, security based on compliance, security based on possible unwarranted trust of third parties and so forth. We can often already capture the nature of other security technologies by formalizing the trust and assumptions under which they are secure. But if some existing solution is based on some possibly unwarranted trust or assumption, when we are integrating with a such a partially flawed legacy technology we should not base the security of our own component on that assumption, even though overall security might formally break when the assumption fails . Having only part of the system depend on the unwarranted assumption might give some kind of graceful degradation of security. There seems to be plenty of room for fundamental research on how to best and most practically model and integrate disparate trust structures and technologies to make systems that are as secure as possible and/or has some optimal graceful degradation of security .

- **Other Non-Security Technology** We also need to be able to integrate with existing technology for building large software systems. For example, how do we make COED technology fit into existing methodologies of enterprise integration possibly across administrative borders and borders with different trust structures, while maintaining some control over the overall security. This seems mainly a technologic challenge. As a last example, how do we fit COED technology that is often developed with precise and rather static trust structures in mind (like, as most half the parties are bad) into pervasive systems where the connectivity and trust structure are constantly changing? This last topic seems to call for more fundamental research.

5.5.2 Compiler and Language Support

If we wish to enable application programmers to create, verify, maintain, and deploy COED applications tuned for performance, correctness, and security, then we need to develop tools to help these programmers. Referring to the analogy above between MPC today and programming in the mid-1950s, we believe that high-level programming languages that abstract and integrate COED capabilities with other computation are required, as are optimizing, verified compilers that generate efficient code that leverages a diverse set of COED technologies. For example a system should work with the best available underlying technology (MPC, FHE, or SSE) which fits the use case, and this choice should be essentially oblivious to the programmer. Specifically, we advocate for four related research agenda items in the COED community:

1. Programming languages with easy to understand semantics that allow reasoning about COED security and performance. Regarding security, Mike Hicks points out that, "one reason to have a high-level language with an understandable semantics is to allow people (or tools) to reason that a computation satisfies their privacy goals". Regarding performance, Rafail Ostrovsky points out that amortization of cost, for example in communication among parties to a computation, has been the focus of significant work from the 90s onward. However, typical COED systems today still have relatively little ability to make useful global amortization decisions

2. Compilers that can consume such languages and automatically generate efficient code in multiple COED paradigms, and can mix such paradigms at will (even within a single target program). Such compilers should not only provide instrumentation to identify bottlenecks, but also include optimization to eliminate them where possible. A diverse set of platform parameters is a difficult consideration for insecure code, and even more so in secure programs. Different solutions may dominate different regimes of operation,

and computation, storage, network, and trust requirements will play a role in this compilation. We do not believe that we can expect to develop specialized compilers for COED technology of the quality provided by commercial compilers for non-COED systems. Could we instead develop COED technology that integrates seamlessly into existing compiler technology?

3. Libraries of "COED-friendly" variants of common algorithms are needed. For example not only basic statistics needed for querying datasets, but and more complex, but widely used, data mining algorithms (e.g. k-means clustering and principal component analysis), as well as other algorithms needed for efficient data processing such as Fourier transforms. These would enable the use of COED technologies in an increasing number of application areas.

4. Formal verification tools equipped to verify COED code equivalence to non-COED (i.e. in the clear) implementations, and to prove properties of COED code; and a verified COED compiler.

5.5.3 Leakage

Leakage can occur in many guises, for example leakage of access patterns of searched data, or leakage of intermediate results in a general computation, or leakage of the control path of an executed program. Many concepts of leakage have been studied previously in the side-channel community; where data leakage corresponds usually to differential power analysis style attacks and control flow leakage corresponds to simple power analysis style attacks, and has been studied in the theoretical community as well.

However, in the context of COED the complexity of leakage is larger and more nuanced. There are two sides of leakage: leakage that we anticipate to permit for the sake of efficiency, and leakage that we anticipate to be resilient against in case that it happens. For example:

- **Access Pattern Leakage:** One can think that security provided by deterministic encryption is not great, but easy to understand. However, the extent of statistical attacks relying on equality leakage does depend on the data and is therefore hard to predict. Therefore , better metrics are needed to have more accessible security statements for SSE proposals that would allow practitioners to pick the solutions suitable for their applications.

  ORAM technology can help avoid data access patterns from leaking, but traditionally it requires a significant practical overhead. Recently, due to theoretical advancements, as well as engineering efforts (optimizing block sizes, using hardware-accelerated ORAM), this technology is advancing in practicality. ORAM not only helps in avoiding leakage of access in database search, i.e. in SSE style applications, but can also be used to help secure general COED applications. For example consider a standard matrix algorithm such as Gaussian Elimination, this is (when implemented correctly) a constant time, branch free algorithm. However, the access patterns into the matrix, in particular row-swaps for pivoting, can reveal partial information on the underlying matrix. Thus providing an ORAM functionality for COED data can enable improved security.

  This improved security comes at a cost, and hence it is important to be able to reason about data access pattern leakage and quantify it. This opens up whole new realms of algorithm analysis, which has barely been touched on in the literature.

- **Data Leakage:** Above we mentioned that the ability to open data in an MPC algorithm enables one to compute more complex functionalities at a reduced cost. This ability has not been pushed to its limit, as it is unclear how any resulting leakage can be controlled. For example it may be the case in a particular algorithm that not all variables need to be computed on securely. Thus analysis tools are need to determine which variables must be kept secure, and which can be revealed, and more importantly when. Thus data leakage can be utilized as an optimization tool within COED technologies.

- **Control Flow Leakage:** In general computation a basic construct is a control flow statement that depends on data, for example the if-then-else constructs and do-while constructs. If the control flow depends on secret data then the execution path of the program will reveal information about the underlying data. When discussing access patterns for data we remarked that using ORAM would prevent leakage of information; we can think of a system which just uses ORAM for data leakage prevention as a "Harvard Architecture" COED system. The natural extension is to produce a "von Neumann Architecture" COED system in which the program data is also assumed to be secure.

  Thus access to the program code can also be secured via ORAM. This leads to an interesting case of how to ensure that two distinct control flow paths are indistinguishable. This is a topic which has been studied in the side-channel community. Incorporating lessons learned in the side-channel community will most likely aid research in the COED community, and could then feed back into more secure side-channel aware compilers for smart cards.

- **Size Leakage:** The sizes of data, the running times of programs, the sizes of information sent over the network has received little attention in terms of leakage. This type of information can be obtained through side-channel or simple monitoring attacks, and yet may reveal information about the underlying program (such as how quickly a program terminated or how many results a query returned). The primary reason these have been less studied is that the problem appears to be difficult to analyze by rigorous means. For example, if one wanted to hide the exact size of a database, one can pad the database, but how many "dummy records" would be acceptable?

5.5.4 Focus on Practicality and Performance

There is a continued need to keep up fundamental research on the underlying technologies like MPC, FHE and SSE. Whereas it is obviously time to begin a focus on technology transfer of COED technology, we believe that the fundamental research chipping away on developing more practical COED technologies also an important effort underplaying the future success of COED technology. We have come a long way since the feasibility results of the 1980s, in some case improving the efficiency by several orders of magnitude, but there is still a long way to go before we can claim the COED technology to be efficient in any conventional sense. Furthermore, we understand very little about the limits on efficiency of COED technology . We therefore believe that we still have a low of important research to do on upper bounds and lower bounds on various measures of efficiency.

Looking a the future of fundamental research on the performance of COED technology, we believe that the research area could benefit hugely by being augmented by a much larger effort focusing on *practical security* and *practical efficiency* of protocols. The research landscape is still too dominated by a preference for theoretical as opposed to practical improvements. This is in part due to the fact that these technologies were considered at first to be impractical, but now, theory has pushed the research into the realm of practicality. The gap needs to be closed by balancing the research, and thus an influx into the practical and engineering side of things is due. In theoretical research on *efficiency* of secure protocols the focus has traditionally been focusing on asymptotic security. Yet, it is often the case that the solution that is asymptotically best is not the best one in practice, not now and possibly not ever in the future. Often the hidden constants are so huge that the promise that the asymptotically good protocol will eventually become more efficient is empty, as this eventuality is so far into the future that the protocol will be surpassed by other and better technologies by the time this eventuality was supposed to materialize.

Creating a much larger research focus on practicality involves a boot-strapping problem of having some of the best researchers in the field actually focussing their research on practicality as opposed to asymptotic efficiency and having the community and the best venues acknowledge and accept research in this direction. There are several ways the focus on practicality could be strengthened. We could develop new models and tools that allow evaluating and comparing the practical efficiency of different protocols. We could also develop standard architectures and benchmarking sets and services allowing to run implementations towards these standard benchmarks. Other related fields like distributed systems are much further developed in these aspects and we might get valuable input from these research communities.

We note that we suggest augmenting the existing methodologies by a focus on practicality. We by no means suggest discontinuing the existing methodology of investigating asymptotic efficiency. But focusing on practical security has it own problems. For instance, it tends to create an over focus on unimportant details like bit fiddling which is mainly of interest for efficient implementation and which are very technology dependent. This means that research on practicality tends to miss more fundamental break throughs. Examples of recent and important practical advancements coming from the research community otherwise mainly focusing on asymptotic security include, Lindells optimal cut-and-choose technology for Yao circuit based protocols, practical OT extension, the TinyOT protocols, the free-XOR trick, garbled row reduction, the LEGO family of protocols, the BDOZ/SPDZ family of actively secure MPC protocols and there are many, many more.

5.5.5 Grand Challenge

We end with a grand challenge for the community: The ultimate challenge when it comes to COED is to enable secure computing on huge data sets. In general, we are still far away from achieving this goal, which is very unfortunate considering that many use cases require such capabilities. Nevertheless, recent years have seen breakthroughs that enable the size of the computation to be much larger than previously thought possible.

In addition, there is an understanding that allowing some leakage is very beneficial and can yield much faster computations (e.g., leaking access patterns). Unfortunately, however, very little is understood about the ramifications of such leakage (what is actually leaked, what can be derived from the leakage, the effect of auxiliary knowledge on the data triangulated with the leakage, and so on). Thus, developing protocols for dealing with massive data, and understanding the consequences of partial leakage in this context, are two primary research goals of the field.