Cloud Security Mechanisms

Christian Neuhaus, Andreas Polze (Hrsg.)

Technische Berichte Nr. 87

des Hasso-Plattner-Instituts für
Softwaresystemtechnik
an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 87

Christian Neuhaus | Andreas Polze (Hrsg.)

Cloud Security Mechanisms

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2014
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2014/6816/
URN urn:nbn:de:kobv:517-opus-68168
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-68168

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-281-0

Cloud Security Mechanisms

Editors: Christian Neuhaus, Andreas Polze

December 17th, 2013

Contents

Introduction

The advent of cloud computing [1, 2] has greatly changed the IT landscape over the recent years: Cloud providers offer a variety of services as on a pay-per-use billing model over the internet. These services are grouped in different categories based on their level of abstraction. The most basic form of these services is described as Infrastructure-as-a-Service (Iaas): They offer service abstractions of hardware resources (e. g. storage space, virtual machines). On a more abstract level, platform-as-a-service (PaaS) provides an execution environment for services using specialized software frameworks. This way, the programmer can concentrate on the application logic while deployment, service scaling and data logistics are solved by the PaaS platform. Finally, entire software products are offered as web applications under the term software-as-a-service (SaaS).

The virtue of cloud computing is the possibility to decouple applications and services from their physical infrastructure: The operation of services is outsourced to cloud providers in large-scale data centers. This leads to a consolidation effect of hardware resources as the varying load on different services can be evened out across the datacenter. For the customer, this way of provisioning resources offers several benefits: The pay-per-use billing model allows provisioning of a large amount of resources on short notice with little up-front investment: Large-scale web applications can be created without building a dedicated datacenter. In addition, the flexible provisioning allows a cloud service consumer to quickly scale his infrastructure depending on his demand.

The greatest challenge of cloud computing is however the question of security: If data and operations are outsourced to a third party, maintaining confidentiality, integrity and availability are a challenge. This is further exacerbated by a complicated legal situation as cloud providers often operate across countries and continents.

Until recently, most security algorithms relied on a clear distinction between trusted and untrusted infrastructure. Software products assumed to be running on trusted infrastructure or heavily relied on the presence of trusted services in a distributed scenario. With cloud computing, the situation is more complex: On the one side cloud service providers depend on their public reputation – they therefore have to provide good service (i. e. *integrity*, *availability*) and ensure data confidentiality. On the other hand, cloud operators cannot rule out the possibility of admin personnel abusing their privileges. While intentional service disruption or data corruption offers little incentive, theft of data is not necessarily discovered and can be monetized. Consequently, data confidentiality is at greater risk than correctness and continuity of service.

It is therefore reasonable to assume a *honest-but-curious* adversarial model for designing security mechanisms to protect data confidentiality in the cloud. This adversarial model assumes that a cloud operator will offer correct and continuous service operation while trying to learn information by eavesdropping and monitoring of program execution.

The current state-of-the-art in access control mechanisms (role-based access control) and cryptography (symmetric and asymmetric encryption) can only partly solve the security challenges of cloud infrastructures. Over several years of research in security and cryptography, novel mechanisms, protocols and algorithms have emerged that offer

new ways to create secure services atop cloud infrastructures. These mechanisms make use of the honest-but-curious adversarial model to increase security of cloud applications.

In the summer term of 2013, the seminar *cloud security mechanisms* was held at the Hasso-Plattner-Institute. This technical report presents student contributions from the seminar.

References

[1] A. Polze, "A comparative analysis of cloud computing environments," Hasso-Plattner-Institute for Software Engineering, Tech. Rep., 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A berkeley view of cloud computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

Threshold Cryptography

Cloud Security Mechanisms Seminar – Summer Term 2013

Björn Groneberg

1 Introduction

In today's world security plays a big role. Every critical system or infrastructure is secured by access control systems and encryption mechanisms. To extend the security in some cases it is not possible that one person or authority alone can access these infrastructures but rather a group of authorities which perform a mutual checking of their behavior. One of the most famous examples is probably the launch of a nuclear bomb where at least two engineers have two use their keys to push the red button. Another example could be the question who is allowed to sign pay checks in a big company. If only one person is responsible then he can misuse this position for his own benefits. To prevent this, pay checks could only be valid if two or more authorities have signed them.

Threshold cryptography mechanisms allow it to perform these tasks in a secure and efficient way. The idea is that secrets should not be programmatically hidden which could lead to a single point of attack but rather using mathematical procedures. The advantage of this approach is, that the global secret is not stored at any place but is shared in a mathematical way. This paper shows how this can be achieved and which problems can occur in the mechanisms.

Starting at section 3 the mathematical and cryptoraphical basis such as modulo operator, Lagrange polynomial interpolation and Elgamal encryption will be presented. The details of the threshold cryptography mechanisms are shown in section 4. This includes an explanation of a secure secret sharing scheme and a threshold encryption scheme. In the end, sections 5 and 6 summarize the report and give an overview about further research questions.

2 Related Work

One of the first publications about threshold cryptography was published by Adi Shamir in 1979. In [1] he describes a technique how to divide data into pieces in a secure way which is still easy to compute. This mechanism is the basis for most of the threshold crypto systems and will be described in detail in section 4.2.

A public key encryption scheme in combination with a signature scheme was published by Taher Elgamal 1985 in [2]. He describes an algorithm which bases on the Diffie-Hellman-Problem and discrete logarithm problem to encrypt and sign messages. A closer look on the Elgamal encryption scheme will be given in section 3.3.

A threshold encryption scheme as a combination of Elgamal cryptosystem and Shamir's secret sharing was published in [3] by Desmedt and Frankel. They proposed a practical non-interactive public key system. The details and problems with combining this mechanisms are also described in this paper in section 4.3. Further remarks on threshold Elgamal was also published in [4].

Another part of threshold cryptography are threshold signatures and a multi-authority election system. Shoup presented in [5] a threshold RSA signature scheme which is completely non-interactive. A universal multi-party election system which guarantees privacy, verifiability and robustness was published by Cramer et. all. in [6]. An overview of both mechanisms will be given in section 4.4.

3 Foundations

In this section different basic techniques which are necessary for threshold cryptography are introduced. At first, basic mathematics like the modulus operator and residue class systems are explained. Afterwards, the Lagrange polynomial interpolation and the Elgamal encryption scheme will be described.

3.1 Basic Mathematics

Mathematical foundations such as the modulo operator and residue classes are presented in this chapter.

Modulo Operator

The modulo operator is a mapping which is defined as

$$f_p : \mathbb{Z} \to \mathbb{Z}_p \tag{1}$$

where $p \in \mathbb{N}$ and $p > 1$. The modulo operator finds the remainder of an integer if it is divided by the number p. If two numbers have the same remainder by the division of a number p they are called modulo congruent to the number p. An example is shown in the following listing.

$$\begin{aligned} & 20 \bmod 6 = 2 \quad \text{and} \\ & 14 \bmod 6 = 2 \\ \Rightarrow \quad & 20 = 14 \mod 6 \end{aligned}$$

Residue Classes and residue class system

If all integers which are congruent to a given modulus are collected to one class, this is called a residue class. Usually, the smallest positive integer in each class is chosen as a representative. A residue class system are all residue classes to a given modulus which are extended with an additive and multiplicative operation. Using this operations, additive and multiplicative inverse elements are defined as follows:

$$\text{additive inverse: } a + (-a) = 0 \mod p$$

where $(-a)$ is the additive inverse element of a and vice versa.

$$\text{multiplicative inverse: } a \cdot (a)^{-1} = 1 \mod p$$

where $(a)^{-1}$ is the multiplicative element of a and vice versa. If the modulus p is prime, every element has a multiplicative inverse element which can be computed using the extended Euclidean algorithm.

3.2 Lagrange Polynomial Interpolation

The idea of the Lagrange polynomial interpolation is to find a polynomial which matches to a given set of points. The graphical representation of this problem is shown in figure 1. A solution for this problem was presented by Joseph-Louis Lagrange in 1795.

If there are $k+1$ data points $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$ where no two x_j are the same, then there exists a polynomial interpolation of the least degree k with the following formula:

$$L(x) := \sum_{j=0}^{k} y_j \ell_j = y_0\ell_0 + \ldots + y_j\ell_j + \ldots + y_k\ell_k \tag{2}$$

The ℓ_j are the Lagrange basis polynomials which are defined as:

$$\ell_j := \prod_{\substack{0 \le m \le k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{x - x_0}{x_j - x_0} \cdot \ldots \cdot \frac{x - x_{j-1}}{x_j - x_{j-1}} \cdot \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdot \ldots \cdot \frac{x - x_k}{x_j - x_k} \tag{3}$$

The Lagrange polynomial interpolation is unique for the least degree k. For example, if

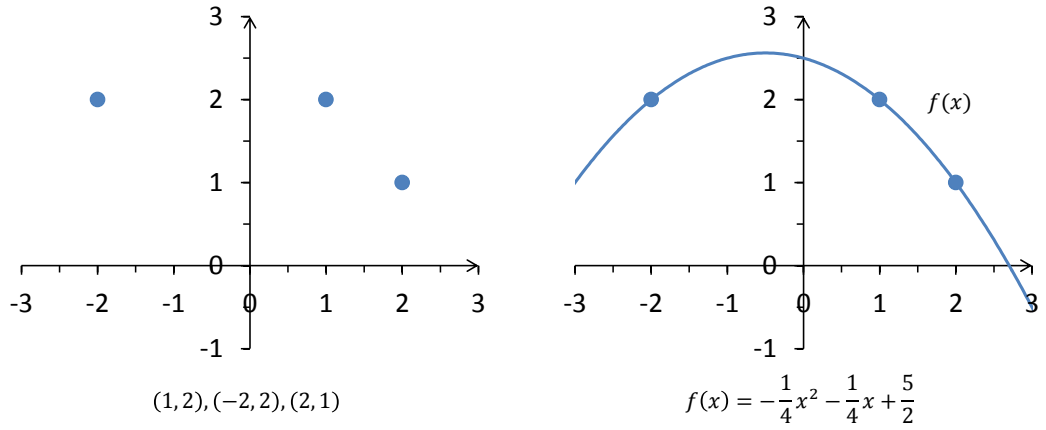

Figure 1: Lagrange interpolation of given points

there are three points $(1,2), (-2,2), (2,1)$, then one can start to calculate the Langrange basis polynomials with (3):

$$\begin{aligned}
\ell_0 &= \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x - (-2)}{1 - (-2)} \cdot \frac{x - 2}{1 - 2} = -\frac{1}{3}(x^2 - 4) \\
\ell_1 &= \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x - 1}{-2 - 1} \cdot \frac{x - 2}{-2 - 2} = \frac{1}{12}(x^2 - 3x + 2) \\
\ell_2 &= \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x - 1}{2 - 1} \cdot \frac{x - (-2)}{2 - (-2)} = \frac{1}{4}(x^2 + x - 2)
\end{aligned}$$

With these basis polynomials, one can compute the Lagrange polynomial interpolation:

$$\begin{aligned} L(x) &= y_0\ell_0 + y_1\ell_1 + y_2\ell_2 \\ L(x) &= 2 \cdot -\frac{1}{3}(x^2 - 4) + 2 \cdot \frac{1}{12}(x^2 - 3x + 2) + \frac{1}{4}(x^2 + x - 2) \\ L(x) &= -\frac{1}{4}x^2 - \frac{1}{4}x + \frac{5}{2} \end{aligned}$$

As a result the only polynomial with a degree of 2 which matches to the given points is $f(x) = -\frac{1}{4}x^2 - \frac{1}{4}x + \frac{5}{2}$.

3.3 Elgamal Encryption

The Elgamal encryption is an asymmetric encryption scheme which was published 1985 by Taher Elgamal in [2]. The keys for this encryption scheme consist of two parts, the public part and the private part. As the name indicates, the public key is known to all communication parties whereas the private key has to be kept secret. If a person B wants to send a message m to person A, B has to encrypt this message with A's public key. The cipher is send to A and only A can decrypt the cipher with the corresponding private key.

The encryption scheme consists of three parts: key generation, encryption and decryption which will be described in detail in the following.

Key Generation

If A wants to create a public private key pair, she has to find a large prime p and a primitive root g. This primitive root is also called generator for the finite group $\mathbb{Z}_p$. Then, A chooses a random number $a \in \{1, \ldots, p-1\}$ and calculates the number A with $A = g^a \mod p$. The public key is the triple pub $= (p, g, A)$ and the private key is priv $= a$.

As one can see, to compute the private key a from the public key (p, g, A), one has to calculate $a = \log_g A \mod p$. This is called the discrete logarithm problem, to which no efficient solution is known today.

As an example, A can choose the prime $p = 23$ and the primitive root $g = 5$. The random chosen number is $a = 6$ Then, she can compute $A = 5^6 = 8 \mod 23$ and the key pair is pub $= (23, 5, 8)$ priv $= 6$.

Encryption

If B wants to send a message $m \in \{0, \ldots, p-1\}$ to A, B knows the public key pub $= (p, g, A)$ and choose a random number $b \in \{1, \ldots, 1-p\}$. Then, he can compute $B = g^b \mod p$ and $c = A^b m \mod p$. The cipher which is send to A is the pair ciph $= (B, c)$.

To decrypt the cipher without knowing the private key a, one has to compute $m = \left(A^b\right)^{-1} \cdot c = \left(g^{ab}\right)^{-1} \cdot c$ which is hard due to the Diffie-Hellman-Problem.

For example, B wants to send the message $m = 12$ to A and chooses a random number $b = 3$. After computing $B = 5^3 = 10 \mod 23$ and $c = 8^3 \cdot 12 = 3 \mod 23$ the cipher is ciph $= (10, 3)$.

Decryption

With the knowledge of the private key a, A can decrypt the cipher (B, c) to obtain the message m. She computes $m = c \cdot (B^a)^{-1} \mod p$, where $^{-1}$ is the multiplicative inverse

element in the residue class ring $\mathbb{Z}_p$. This inverse element can be computed with the extended Euclidean algorithm. In the example A computes $m = 3 \cdot \left(10^6\right)^{-1} = 3 \cdot 6^{-1} = 3 \cdot 4 = 12 \mod 23$.

The general idea of the Elgamal encryption scheme is the Diffie-Hellman-Problem [7], which says if there are two numbers g^a and g^b, no efficient way is known to compute g^{ab}. In the Elgamal encryption scheme the numbers $A = g^a$ and $B = g^b$ are known but the number $A^b = g^{ab} = B^a$ is not known to a third party but is needed to compute the multiplicative inverse element for the decryption.

4 Mechanism Details

This section describes different mechanisms for threshold cryptography. Starting with basic definitions about secret sharing in part 4.1, part 4.2 explains an algorithm for sharing secrets in more detail. In part 4.3 it is shown how the Elgamal encryption scheme can be combined with a threshold approach. After this, part 4.4 gives an overview about further threshold cryptography mechanisms.

4.1 Secret Sharing

The general idea of secret sharing is to distribute a secret s to n parties in that way, that only a subset with a pre-defined amount of parties is able to recompute the secret. In a simple scenario a so called trusted dealer is used to create the secret s with the sub secrets $s_0, s_1, \ldots, s_{n-1}$, which are shared among the n parties. After the creation and sharing phase, the trusted dealer should delete the generated secret s. This means there must be no place, where the global secret s is stored. Even if this secret s is encrypted and protected with an access control system, the storage of s would lead to a single point of attack. The advantage of secret sharing is the possibility to hide the secret mathematically.

There are two possibilities for recomputing the secret. The first one is the (n, n)-threshold, which means that all n parties are necessary to recompute the secret. If there are less then n parties, they must not be able to recompute the secret and every party or group of parties should not be able to retrieve any information about the global secret s from their own secret(s).

The other possibility is the (k, n)-threshold. This means, that a subset of at least k parties is necessary to recompute the global secret. The number k has to be defined during the creation of the secret. But it is not defined which parties have to participate in the computation. Again, it must not be possible that subsets with less than k parties are able to recompute the secret or have any information about it.

Simple Approach

A very simple but also a very insecure approach for secret sharing would be the splitting of the secret. Figure 2 shows an illustration how the 20 digit key is split into 5 sub keys with 4 digits each. One disadvantage of this technique is the fact that a (n, n)-threshold is needed. The more significant disadvantage is the dramatically reduced key range if 4 of 5 parties are combining their keys. For every party 4 out of 20 digits are known and the key space would be 10^{16}. But if 4 parties combining their keys, 16 digits are known and the key space would be 10^4.

After explaining this simple approach for sharing a secret, the next section gives a detailed description of a more sophisticated technique.

Figure 2: Naive solution for secret sharing

4.2 Shamir's Secret Sharing

This secret sharing algorithm was published by Adi Shamir in 1979 in [1]. It is a (k, n) threshold sharing scheme and is based on the Lagrange polynomial interpolation which is described in section 3.2. Shamir's secret sharing consists of two part which are described in the following.

Dealing Algorithm

In the beginning, the parameters k and n for the (k, n)-threshold and the global secret $s \in \mathbb{Z}_q$ have to be defined. The value q has to be a prime, because only in prime residue class rings it is ensured, that every element has a defined multiplicative inverse element which is necessary for computation.

The first step is to choose $k-1$ coefficients $a_1, \ldots, a_{k-1}$ and set $a_0 := s$. Then, a polynomial $f(x)$ is built with $f(x) = a_0 + a_1 x + a_2 x + a_{k-1} x^{k-1}$. To create the shares, a variable i is set to $i = 1, \ldots, n$ and points with $s_i = (i, f(i)) \mod q$ of the polynomial $f(x)$ are computed. In the end, every party gets at least one point s_i as a secret share, whereas the first part i of the share can be public and the second part $f(i)$ has to be kept in private. The following shows an example of creating shared secrets.

$$(3, 5)\text{-threshold}, \quad q = 11, \quad s = a_0 = 6$$

Choose random $k-1$ coefficients and build polynomial $f(x) = a_2 x^2 + a_1 x + a_0$:

$$a_1 = 1, a_2 = 3$$
$$f(x) = 3x^2 + x + 6$$

Calculating shared secrets $s_i = (i, f(i)) \mod q$ with $i \in \{1, \ldots, n\}$:

$$s_1 = (1, 10), s_2 = (2, 9), s_3 = (3, 3), s_4 = (4, 3), s_5 = (5, 9)$$

This dealing phase is usually performed by a trusted dealer who has to forget all information about the creation especially the global secret s and the shares s_i to guarantee a real shared secret without a single point of attack.

Recomputation

To obtain the global secret s from the shared ones, k points $s_i = (x_i, y_i)$ are needed. The goal in the recompuation step is to find the polynomial $f(x) = a_0 + a_1 x + a_2 x + a_{k-1} x^{k-1}$ with $f(0) = a_0 = s$ which was created during the dealing phase. This can be done by using the Lagrange interpolation with $f(x) = L(x)$ for the k points. Due to the fact that only the value $f(0)$ has to be computed the Lagrange basis polynomials (3) can be

simplified by setting $x = 0$ and $S \subseteq \{1, \ldots, n\} \| S \| = k$ with S as the subset of parties which participate in the recomputation.

$$\ell_{j,0,S} := \prod_{\substack{m \in S \\ m \neq j}} \frac{-x_m}{x_j - x_m} \mod q \tag{4}$$

In the end, the global secret can be computed with the Lagrange formula (2) using the basis polynomials from (4)

$$s = L(0) = \sum_{j \in S} y_j \ell_{j,0,S} \mod q \tag{5}$$

Due to the uniqueness of the Lagrange interpolation to obtain a polynomial with degree $k-1$ from k points it does not matter which parties are involved in the recomputation step, as long as there are exactly k shares. The following listing shows an example how the global secret can be recomputed using the Lagrange interpolation.

$$\begin{aligned}
\ell_{1,0,\{1,2,3\}} &= \frac{-x_2}{x_1 - x_2} \cdot \frac{-x_3}{x_1 - x_3} = \frac{-2}{1-2} \cdot \frac{-3}{1-3} = 6 \cdot 2^{-1} = 6 \cdot 6 = 3 \mod 11 \\
\ell_{2,0,\{1,2,3\}} &= \frac{-x_1}{x_2 - x_1} \cdot \frac{-x_3}{x_2 - x_3} = \frac{-1}{2-1} \cdot \frac{-3}{2-3} = 3 \cdot (-1)^{-1} = 3 \cdot 10 = 8 \mod 11 \\
\ell_{3,0,\{1,2,3\}} &= \frac{-x_1}{x_3 - x_1} \cdot \frac{-x_2}{x_3 - x_2} = \frac{-1}{3-1} \cdot \frac{-2}{3-2} = 2 \cdot 2^{-1} = 2 \cdot 6 = 1 \mod 11
\end{aligned}$$

$$\begin{aligned}
s &= L(0) = y_1 \cdot \ell_{1,0,\{1,2,3\}} + y_2 \cdot \ell_{2,0,\{1,2,3\}} + y_3 \cdot \ell_{3,0,\{1,2,3\}} \mod q \\
s &= 10 \cdot 3 + 9 \cdot 8 + 3 \cdot 1 \mod 11 \\
s &= 6
\end{aligned}$$

Again, after the recomputation which should be perfomed by the trusted dealer, he has to forget everything to maintain the security of this mechanism.

Evaluation

With Shamir's secret sharing algorithm the global secret s is mathematically hidden with the shared secrets s_i. No party is able to retrieve any information about the global secret if there are less than k shares available. It can be proved, that if one share is missing, every possible solution for the global share s has the same probability. Figure 3 shows a graphical representation of this fact. The points $x = 1$ and $x = 2$ are known with the values but the point $x = 3$ is not known in the $(3, n)$-threshold.

Furthermore, Shamir's secret sharing algorithm is very flexible. It is easily possible to increase the overall number of parties n after the initial dealing phase while compute new shares from the polynomial $f(x)$. It is also possible to remove shares, but only if they are really destroyed. Another advantage of this mechanism is the easy replacement of all shares without changing the secret. A trusted dealer just needs to find a new polynomial $f^*(x)$ with the same a_0. Of course, the old secrets have to be destroyed, as well. At last, it is also possible that one party can obtain more than one share. This could be useful if one party should be able to recompute the secret alone while other parties should not.

A disadvantage of this algorithm is that it has to be ensured that the trusted dealer is really trustworthy and deletes every information about the shared and the recomputed secret.

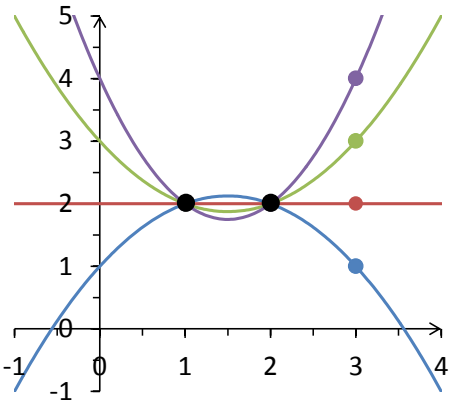

Figure 3: Interpretation of Shamir's secret sharing

4.3 Threshold Elgamal Encryption

The Elgamal encryption scheme can be used in threshold environments where the private key is shared over n parties. To achieve this, the Elgamal encryption scheme which was described in section 3.3 is combined with Shamir's secret sharing algorithm from the previous section.

Key Generation and Encryption

The generation of the private and public key is the same algorithm like the normal Elgamal encryption. The difference is that the secret key is shared using Shamir's secret sharing.

It is important to have a closer look on the values which are used for the modulus in both algorithms. To achieve a secure encryption, the Elgamal encryption makes use of the prime p for the modulus. When combining this scheme with Shamir's secret sharing the modulus q for the secret sharing has to be set as $q = \frac{\varphi(p)}{2}$ where φ is Euler's totient function which is $\varphi(p) = p - 1$ if p is prime. It should also be remembered that the modulus q for Shamir's secret sharing has to be a prime. The reason for this constraint between p and q and the problem with this will be shown in the decryption part of the threshold Elgamal encryption scheme.

The encryption for the threshold scheme is a normal Elgamal encryption with the public key and a random chosen number. The advantage is that the party who encrypts the message cannot see that a threshold scheme is used and the information about the parties who share the secret is hidden.

Decryption

After receiving the encrypted message, at least k parties have to work together to decrypt it. A simple approach would be, that every participating party i sends his share s_i to a trusted dealer and this dealer recomputes the global share which is the private key for decrypting the cipher. The disadvantage is that the global share is computed during the encryption and the parties have no control about the further use of their shares s_i.

To improve the decryption different sources state the following algorithm. Every participating party j receives the cipher (B, c) and computes a decryption share

$$d_j = B^{y_j} \mod p \tag{6}$$

This decryption share is send to the trusted dealer who computes the message m with the following formula:

$$m = \left(\prod_{j \in S} d_j^{\ell_{j,0,S}} \right)^{-1} \cdot c \mod p \tag{7}$$

Where S is the subset of parties which participate in the decryption and $\ell_{j,0,S}$ are the Lagrange basis polynomials from formula (4). The following listing shows how this formula could be derived from the Elgamal decryption scheme:

$$\begin{aligned}
m &= (B^a)^{-1} \cdot c \mod p \\
\Rightarrow m &= \left(B^{\sum_{j \in S} y_j \ell_{j,0,S} \mod q} \right)^{-1} \cdot c \mod p \\
\overset{*}{\Rightarrow} m &= \left(B^{y_0 \ell_{0,0,S} \bmod q + \ldots + y_{k-1} \ell_{k-1,0,S} \bmod q} \right)^{-1} \cdot c \mod p \\
\Rightarrow m &= \left(B^{y_0 \ell_{0,0,S} \bmod q} \cdot \ldots \cdot B^{y_{k-1} \ell_{k-1,0,S} \bmod q} \right)^{-1} \cdot c \mod p \\
\Rightarrow m &= \left(\prod_{j \in S} B^{y_j \ell_{j,0,S}} \right)^{-1} \cdot c \mod p \\
\Rightarrow m &= \left(\prod_{j \in S} d_j^{\ell_{j,0,S}} \right)^{-1} \cdot c \mod p
\end{aligned}$$

Constraints for Decryption

The implication from the previous section which is marked with * is only valid under certain conditions. The term

$$B^{y_0 \ell_{0,0,S} \bmod q + \ldots + y_{k-1} \ell_{k-1,0,S} \bmod q} \mod p$$

can also be written as

$$B^{k \cdot q + a} \mod p$$

This is possible because the operator mod q can only be applied to each summand but not on the whole sum in the exponent. If the sum exceed the modulus q there are two possible cases. In the first case k is even and with $2q = p - 1$ it follows:

$$B^{\frac{k}{2} \cdot 2q + a} = B^{\frac{k}{2} \cdot p - 1 + a} = B^{\frac{k}{2} \cdot p - 1} \cdot B^a = B^a \mod p$$

In this conversion a direct implication from Fermat's little theorem is used, which states that $a^{p-1} = 1 \mod p$ if p is a prime. In the other case k is odd and it follows:

$$B^{\frac{k-1}{2} \cdot 2q + q + a} = B^{\frac{k-1}{2} \cdot p - 1 + q + a} = B^{\frac{k-1}{2} \cdot p - 1} \cdot B^q \cdot B^a = B^q \cdot B^a \neq B^a \mod p$$

This means, that the marked implication is only valid if the k in the sum is even which has a theoretical probability of 0.5. As a result of this investigation it can be claimed that using threshold Elgamal encryption in this way, false results can be computed. Unfortunately, it is not clearly visible which value the sum has, because it is never directly computed.

The following listing shows an example where the decryption of threshold Elgamal fails due to the previous explained problem.

$$\begin{aligned} \text{pub} &= (23, 5, 8) \\ \text{priv} &= 6 \\ \text{shares} &= (1, 10), (2, 9), (3, 3), (4, 3), (5, 9) \end{aligned}$$

The message $m = 12$ is encrypted with the public key and the random chosen $b = 3$:

$$\begin{aligned} B &= 5^3 = 10 \mod 23 \\ c &= 8^3 \cdot 12 = 3 \mod 23 \\ \text{cipher} &= (10, 3) \end{aligned}$$

The decryption starts with computing the decryption shares d_j for each party and the Lagrange basis polynomials $\ell_{j,0,S}$ from the example in section 4.2

$$\begin{aligned} d_1 &= 10^{10} = 16 \mod 23 \\ d_2 &= 10^9 = 20 \mod 23 \\ d_3 &= 10^3 = 11 \mod 23 \end{aligned}$$

$$\ell_{1,0,S} = 3, \ell_{2,0,S} = 8, \ell_{3,0,S} = 1 \text{ with } S = \{1, 2, 3\}$$

The trusted dealer should compute the message with the formula (7):

$$\begin{aligned} m' &= \left(d_1^{\ell_{1,0,S}} \cdot d_2^{\ell_{2,0,S}} \cdot d_3^{\ell_{3,0,S}}\right)^{-1} \cdot c \mod p \\ m' &= \left(16^3 \cdot 20^8 \cdot 11^1\right)^{-1} \cdot 3 \mod 23 \\ m' &= (17)^{-1} \cdot 3 \mod 23 \\ m' &= 19 \cdot 3 \mod 23 \\ m' &= 11 \neq 12 = m \end{aligned}$$

It is visible that the decrypted message m' is false. This could be corrected by multiplying with the multiplicative inverse element of the factor $B^q = 10^{11}$:

$$\begin{aligned} m &= (B^q)^{-1} \cdot m' \mod p \\ m &= \left(10^{11}\right)^{-1} \cdot 11 \mod 23 \\ m &= 22 \cdot 11 \mod 23 \\ m &= 12 \end{aligned}$$

To circumvent these constraints, one could theoretically set $q = p - 1$ to use the modulus in the exponent in the right way. But with the constraint that p and q have to be prime the amount of possible values for these primes would be very small. Another solution is to detect if the decrypted message has to be multiplied by the multiplicative inverse element of B^q and correct the message in that way if necessary.

4.4 Further Threshold Cryptography Mechanisms

Beside the Elgamal encryption there exists further cryptography mechanisms which can be extended with a threshold implementation. In this section an overview of a threshold RSA signature scheme and a distributed e-voting mechanism is displayed.

Threshold RSA

The RSA signature scheme is used to verify that a message was send from a specific sender and to ensure the message integrity. It requires a public-private key pair and a cryptographic hash function $H(x)$. If B wants to sign a message m, he computes the Hash $H(m)$ of this message and encrypts this with his private key.

$$\text{sign} = \text{enc}\left(H(m), \text{priv}\right)$$

If A receives the signed message m' she can verify the signature by decrypting the signature with B's public key and computes the Hash $H(m')$ of the received message. If the decrypted hash and the computed hash are equal she can be sure that B sent this message and the message was not manipulated during the transmission.

$$\text{dec}(\text{sign}, \text{pub}) \stackrel{?}{=} H(m')$$

This signature scheme can be extended with a (k, n)-threshold mechanism. The private key is shared over n parties where k parties have to work together to create a valid signature. Every party i which participates in the signature creation computes a signature with his own shared private key.

$$\text{sign}_i = \text{enc}\left(H(m), \text{priv}_i\right)$$

A trusted dealer receives all partial signatures and collect them to retrieve the global signature.

$$\text{sign} = \text{collect}\left(\text{sign}_1, \ldots, \text{sign}_k\right)$$

The mathematical basis for this mechanism is very similar to the threshold Elgamal encryption. In the end, the receiver of the signed message can verify this threshold signature in the normal way. Usually the receiver does not even know that a threshold signature scheme was used. A detailed description of threshold signature schemes can be found at [5].

E-Voting

In an electronic voting system the votes could be encrypted to guarantee anonymization and the integrity of the vote. The count of the votes should be observed that the counting authority is not able to manipulate the vote. To achieve this, threshold cryptography can be used as well. Cramer presented a multi-authority election system in [6] which will be described in the following.

The votes are represented as numbers, more precisely -1 for a negative and $+1$ for a positive vote. These votes are encrypted with the public key of the counting authorities using the Elgamal encryption scheme. After the encryption the votes are published on a public bulletin board which ensures that every voter can verify that his vote has not been manipulated. Due to the random part in Elgamal encryption similar votes do not result in the same encrypted vote. This means that everybody could see *that* somebody has voted but it is not visible *what*.

After publishing the encrypted vote one or more authorities counting these votes and compute a encrypted result of the election. Mathematical procedures which are similar to these of the threshold Elgamal encryption are used to guarantee that the counting can be performed even if the votes are encrypted. After counting, the authorities use their shared private keys to decrypt the voting result. The sharing ensures, that one authority alone is not able to manipulate the election.

5 Future Work

Threshold cryptography is still an open point for further research. One of the most discussing points are the use of the trusted dealer. He is a theoretical point of attack because he knows the global secret during the creation and also the encrypted message. The question is how to implement such an authority in a trustworthy way. One solution could be the distributed generation of the shared secret. In this scenario every participant computes one part of the key in a way that every party receives their shared secret but the global key is never computed directly.

Another open point is the problem of choosing the right modulus for combining the Elgamal encryption scheme with Shamir's secret sharing algorithm. As stated in section 4.3 the choice of the primes p and q which is often claimed as $p = 2q + 1$ can lead to false results. A reason for this could be that the threshold encryption schemes are not very common in real world software solutions. Nevertheless, there are some prototype implementations which can be found at [8] but they use a different approach for the key generation and circumvent the problem with the modulus in the exponent.

6 Conclusion

This paper gave an introduction to threshold cryptography and showed some of the mathematical basis for the used algorithms. The Lagrange polynomial interpolation and the Elgamal encryption scheme are introduced as a tool for threshold cryptography mechanisms. After giving a short overview about general secret sharing and a simple example, the more sophisticated approach of Shamir's secret sharing algorithm was presented which uses the Lagrange polynomial interpolation as basis.

It has been shown how the Elgamal encryption scheme could be extended with Shamir's algorithm to retrieve a threshold encryption mechanism where private keys are shared over different parties. A mathematical approach and a numeric example illustrated the difficulties when combining these two mechanisms. In the end an overview about further threshold cryptography mechanisms such as threshold signatures and multi-authority e-voting was given. As a result, threshold cryptography is a high potential technique in nowadays security environment but has not become widely known in the daily business.

References

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Information Theory, IEEE Transactions on*, vol. 31, no. 4, pp. 469–472, 1985.

[3] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in Cryptology—CRYPTO'89 Proceedings*. Springer, 1990, pp. 307–315.

[4] G. Christoph and A. Steffen, "Secret sharing & threshold decryption mit dem elgamal kryptosystem," 2010.

[5] V. Shoup, "Practical threshold signatures," in *Advances in Cryptology—EUROCRYPT 2000*. Springer, 2000, pp. 207–220.

[6] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[7] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, vol. 67, no. 3, pp. 397–427, 1979.

[8] A. M. Davis, D. Chmelev, and M. R. Clarkson, "Civitas: Implementation of a threshold cryptosystem," 2008.

The Bitcoin Protocol

Cloud Security Mechanisms Seminar – Summer Term 2013

Johannes Henning, Robin Schreiber

1 Introduction

This paper describes the concepts and their basis as well as the implementation of the Bitcoin Protocol. The Bitcoin Protocol is an abstract protocol which tries to achieve byzantine fault tolerance in a distributed system without relying on a central authority that controls all the ongoing activity in the network. In order to solve this problem the protocol introduces a special data structure that all nodes in the network participate in. The participation involves a significant amount of computational power, so that each node has to invest a certain amount of work to be part of the system. The main idea behind this concept is, that malicious or fraudulent contributions to the network do not propagate because it may later on be discovered by other "honest" nodes and cause the hard earned contribution to be rejected by the network. The assumption is, that the majority of nodes are honest and also hold the majority of computing power. The main benefit from this approach is, that one has instantiated a system of trust that is not a single entity but rather distributed across all nodes in the network. A central trusted authority, which approves every participation (but might also exploit the system together with its users) is therefore unnecessary. Since the introduction of Bitcoin as a monetary union, the Protocol has often been directly connected to the Currency, although it is actually an entirely independent concept that can be applied to many different domains.

2 Foundations

A hash function is any algorithm that maps data of variable length to data of fixed length. This implies that there exist multiple inputs that map to the same output e. g. the same hash. This is called a hash collision [1]. All cryptographically save hash functions guarantee that hash collisions are almost impossible to find. If an implementation becomes vulnerable in this regard it will no longer be used for implementing any security related algorithms.

"Public/Private Key Authentication and Encryption" [2] relies on a public key associated with a certain individual and private key only known to that one individual. The algorithms involved allow information to be encrypted with the public key in a way that it can only be decrypted by the private key, as well as the other way around, which is called signing.

3 History

There are three principles Bitcoin is based heavily upon: Public/Private Key Authentication and Encryption, Proof of Work and Time-Stamping.

3.1 Public/Private Key Authentication and Encryption

This is a well-known principle that was first introduced by Merkle [2] as well as further improved. Other publications have discussed this approach in depth and it seems to be well understood in the community, especially considering it is the base for RSA and therefore our everyday SSL. Hence we will not discuss it further in this paper.

3.2 Proof of work

proof of work (POW) was first introduced by Dwork and Naor [3] although they called it"pricing function". Their initial idea was refined and expanded in multiple papers. They identified, that the reason why we get more spam via email than via regular mail is, that emails are significantly cheaper. They wanted to introduce postage for email without hindering the free communication that the whole idea of the internet is based upon. The idea they came up with was to have a user or rather the user's computer solve a mathematical problem that was known to take a certain amount of time, therefore limiting the amount of interactions the user could make with the system e. g. the mail-server. The postage in this case would be the computational time invested for solving the problem. A downside of this approach was identified in their publication as well, namely the utter uselessness of the actual computation. Since a significant amount of time was invested into computing the result of the pricing function it is just plain wasteful to disregard the result afterwards. To this day this is the biggest problem of POW functions.
Jakobsson and Juels [4] summarized state of the art vocabulary and useful implementations considering POW and offered formal definitions. Furthermore they introduced "Bread Pudding Protocols" as a solution for the problem of the useless computation done by POW functions. The name is based upon the reuse of old bread in some eastern dishes and accordingly tries to reuse the computation done by the POW functions. In the Bread Pudding Protocols any server is also a client towards a different server, therefore the server could redistribute smaller parts of the POW that he has to do to his clients, who in turn proof to him that a certain amount of time was invested by solving the small part. The server could then combine the partial solutions of its clients to produce the POW required by its server.
Juels and Brainard [5] considered the dynamic reconfiguration of proof of work function difficulty in order to make DDOS attacks less feasible. When a server comes under an attack it starts to distribute puzzles to its clients, that the clients have to solve before their request gets answered. The difficulty of the puzzle can be scaled according to the amount of requests that the server is facing. Considering the nature of depletion attacks (namely SYN flooding) they also showed a way to implement their solution stateless and therefore not requiring any memory to be allocated on the server for the session before the client has solved its puzzle.
Rivest, Shamir and Wagner [6] looked at POW from a different perspective. They discussed the possibility of releasing documents to the public while guaranteeing that said documents could not be read before a certain amount of time had passed. One way this could be done was by using an encryption scheme that had been compromised, therefore no longer being secure, but still taking a certain amount of time to crack. Since they had real time in mind and not computational time they also introduced the idea of algorithms for POW functions that cannot be parallelized, we will discuss this approach later on.

The suggestion of a concrete algorithm and implementation that the Bitcoin Protocol uses is based upon the aforementioned approaches: Hashcash [7]. Here partial hash-collisions are used to create easy to check but hard to solve POW functions. Hashcash was designed as a measure for metering of publicly available online resources. They also considered multiple versions of their function, relevant to different use cases.

3.3 Time-Stamping

The story of time-stamping is somewhat simpler since most of it is based on the work of Haber and Stornetta. They first introduced the time-stamping of digital documents in 1991 [8] under the light of digital documents being introduced into the legal domain and therefore being dependent on some kind of reliable, not forgeable time-stamp. In 1993 they refined the approaches [9] to cover distributed time-stamping in an unreliable network. This approach is widely implemented and almost verbatim described in the original Bitcoin paper [10].

4 Mechanism Details

In the following two sections we will discuss in depth how POW and time-stamping work and are described in the aforementioned publications.

4.1 Proof of work

Proof protocols are a foundation for many concepts in IT security. While typically used to proof that one possesses a certain secret (e. g. Public/Private Key Authentication), POW functions aim to proof that a certain amount of computational power was spent in a certain amount of time. The domain in which they were proposed was mainly access-metering and abuse-protection. The first use case for POW functions was spam prevention for emails [3] and was based on solving small cryptographic functions on the email header and timestamp. As with all POW functions, solving one is always done through some kind of brute force approach e. g. trying to crack a small encryption or continually modifying a string in order to produce a certain hash collision.
As it is usually the case in cryptography, the POW approach is based on the assumption, that one way functions exist, which is yet to be proven [4]. The generally accepted definition of one way function is, that it is impossible to reverse. Therefore while computation of $f(x)$ given x should be moderately easy, given $f(x)$ it should be multiple magnitudes harder (e. g. infeasible) to compute x. A hash-function is a function that satisfies this criteria, for it is considered to be infeasible to find hash-collisions in all widely used implementations, as soon as vulnerabilities are found in this regard, the usage of the specific implementation is no longer recommended.
There are multiple characteristics a POW function might have:

- probabilistic cost: as opposed to fixed cost, POW functions typically have probabilistic cost. We can further differentiate between bounded and unbounded probabilistic cost, which refers to the predictability of the amount of work a certain POW function will take
- interactive/noninteractive: the user might choose his own challenge to compute or the server might provide one for him. Security implications are obvious, since it would typically be easier for the user to precompute answers, if he is the one choosing them

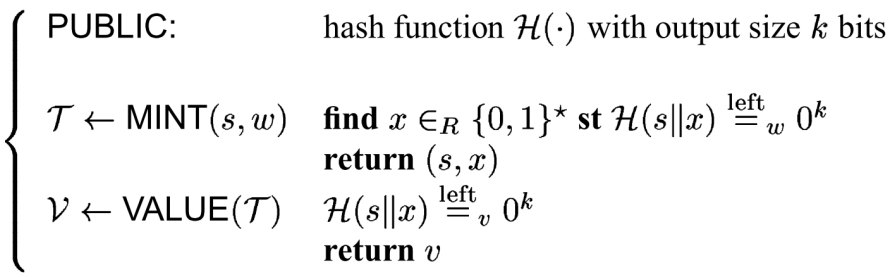

Figure 1: Hashcash pricing function [7]

- publicly auditable: the POW can efficiently be verified by a third party
- trapdoor free: in this context a shortcut within the POW function is called a trapdoor. In early implementations it was considered a good idea for certain use-cases, e. g. having a trusted mail-service that could send mails without investing as much computational power

[3, 7, 4]
While POW functions can be based on any one-way function, the implementation of Hash-Cash and therefore the Bitcoin Protocol, relies on partial hash collisions, is non-interactive, trapdoor free, has unbounded probabilistic cost and is publicly auditable.

As shown in figure 1, the MINT function takes a string s and a number w which is the required size of the hash-collision. It concatenates s with a random binary number x of a specified length and hashes the result to produce a hash-collision of size w with 0^k e. g. the first w characters of the hash need to be zeros. The fastest known algorithm for this MINT function is brute force (e. g. trying out different values for x), there is no telling how fast it will be or if it will ever find a solution (unbounded probabilistic cost) and there is no know trapdoor.
The function returns the original string s and the random addition x producing the hash-collision, therefore any party that acquires the s and x can check the correctness of the values by concatenating x to s and applying the hash-function once. The amount of effort required for this check is therefore significantly lower than the amount of work invested in the minting process, in consequence this MINT function is publicly auditable.
In this particular implementation the string s is chosen by the user (non-interactive), however this does not mean that there cannot be any further requirements on the chosen string. For example, the receiver of the results might require the string to contain a current time-stamp. This approach is similar to the initial idea of Dwork and Naor [3] and is susceptible to a precomputation attack. Fortunately this does not pose a problem for Bitcoin which will become clear further down.

4.2 Time-Stamping

Often there is a need to certify when a document was created or last modified. For instance when documenting the approval of a patent, it is of vital importance to be able to proof when a certain idea was first documented. Hence there exist authorities mainly concerned with verifying analog documents such as patents or contracts. Since analog documents are bound to their medium it is possible to verify that the time-stamped document has not been tempered with and that the stamp by the authority is genuine.

In the age of the internet, digital documents are steadily becoming the norm and it is not hard to imagine a world, where most documents are exchanged in digital form. For this very reason it is paramount to provide an equivalent mechanism for time-stamping and signing digital documents that is infeasible to manipulate.
So how can we achieve this when digital documents are by nature easy to manipulate and not bound to any physical medium? First of all we need to realize what is needed to securely date an event:

> *If an event was determined by certain earlier events, and determines certain subsequent events, then the event is sandwiched securely into its place in history.* [9]

With this idea of linking every event to events that happened earlier and familiar concepts such as public/private key encryption/signing as well as hash functions Haber and Stornetta [6, 9] proposed a distributed signing authority that can securely timestamp documents without the need for making them publicly accessible, which is important for sensitive data.
The basic idea is best illustrated by the following formula:

$$C_n = (n, t_n, ID_n, y_n; L_n) \qquad L_n = (t_{n-1}, ID_{n-1}, y_{n-1}H(L_{n-1})) \text{ [6]}$$

C_n is the client certificate it contains the sequence number n, the time t, the client id ID, the hash of the signed document y and a reference to the last entry L, which in turn has its own time, ID, document hash and the hash of the previous L: $H(L_{n-1})$. So what is being built here is a linear list, with each entry referencing its predecessor. Important to note is also that we save the hash of our predecessors L which in turn contains the hash of its predecessor and so on. It therefore becomes infeasible to insert anything into this list or delete anything from it, since the entire following list would need to be modified, since all subsequent hashes would change. This of course is conditioned on the fact, that we cannot produce two different documents that have the same hash, which is an assumption about the used hash function.
While it demonstrates the taken approach quite well, a linked list is not a particularly efficient way to store huge amounts of data, especially when keeping in mind how one would approach verifying the timestamp of a certain document, which would involve traversing the list so far backward till one reaches a trusted event, while recomputing all hashes up to this point and comparing them with the stored ones. At least when thinking of a globally accessible signing service one would not expect a linked list to be the data structure of choice, which is what Haber and Stornetta identified as well [9].
As a more scalable and distributable approach they suggested storing the certificates inside a Merkle tree [11]. The basic idea stayed the same, only now the list of document hashes is by a binary tree as illustrated in figure 2 [12]. The approaches from both papers are implemented verbatim in Bitcoin, transactions are hashed via Merkle trees and Bitcoin blocks are linked in a linear list.
After a predetermined amount of documents the last round value will be published via a print medium, e. g. newspaper and therefore anchoring the event securely in history.

5 Implementation

As hinted in the Introduction, the Bitcoin Protocol tries to store bits of information in a distributed fashion. At the same time it guarantees the authenticity of the stored information without relying on a central authority that functions as a“trusted part”. In

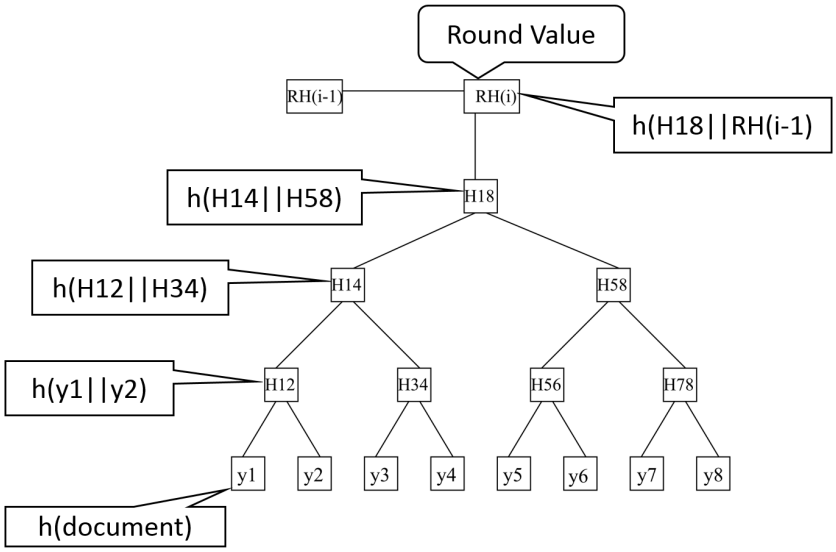

Figure 2: Time-stamped document-hashes are accumulated by a Merkle tree and contained within the round value node via hashing the concatenation of the child node hashes recursively

order to achieve the authenticity of the stored information, the protocol tries to establish something called "stabilizing consensus"[13] with regard to the correctness of the information which is defined as follows: Let us assume that we have different instances of the same information floating around in our network (which in a distributed system is very likely the case). The problem is, that only one instance of the information is correct, while all the others may have been introduced by fraudsters who want to benefit from the propagation of the false information. The Bitcoin Protocol now guarantees that under the assumption that the majority of computing power is in the hands of "honest nodes", the network will always agree on the correct instance of the information.
The Bitcoin Protocol is based on the idea that the information is stored inside Blocks which are then arranged in a chain, where each block is dependent on the hash of its predecessor. This is basically the application of a Time-stamping Server [9, 8] described in the previous sections. The second major ingredient to the protocol is to fuse every block in the chain with a Proof of Work as described in section 4.1, so a reasonable amount of computing power has to be spent on the creation of such a block. In the Bitcoin Protocol the Proof of Work is built up as follows: Every block consist of the information that is stored and a nonce, which is an integer that can be varied arbitrarily. Also included in the block is the hash of the previous block (this implements the time-stamping). The hash is computed by applying the SHA-256 function twice. The protocol now requires the hash of the newly computed block to have exactly k trailing zero bits (similar to the constraint used in the implementation of Hashcash [7]). Again, the number k works as a parameter for the difficulty of computing a new block in the chain.
Each node in the network now functions as a worker, who computes new blocks for an existing chain. In order to do that a node not only tries to find the required hash-collision but also verifies that the information stored in the block is actually correct. Nodes also check the blocks that other nodes have computed for correctness in order to consider them as the basis for the next block. As we assume that the majority of nodes can be considered "hones", correct blocks will sooner or later propagate to all areas of the

network. It is safe to say that this is kind of a chaotic state of the system, as many nodes might compute a valid solution for the block at the same time, and different clusters of nodes might start to consider different versions of the chain as the correct one. As a result a lot of differently sized chains are propagated in the network. To solve this problem, the protocol now defines that the longest chain is the one who stores the correct instance of the information. A node that starts to extend an existing chain will therefore only do that on the longest one, as it behaves according to the protocol. In case a longer chain appears while it has already started computing a block, it will cancel computation and commence extending the new longer chain. Given the assumption that the majority of computing power is in the hands of "honest nodes" who verify the information in a trusted way, the longest chain must therefore also correspond with the correct instance of information. This has a self-stabilizing effect on the network, as it gets less and less likely for the longest chain to be overtaken by another chain, the higher the lead of the longest chain is. In fact the complexity of computing an alternative chain increases exponentially as shown in [10]. This makes critical race-conditions between multiple chains extremely rare cases, which have only been encountered in scenarios where the protocol received a critical update which still needed time until it propagated to all nodes in the network.

5.1 Applications

It is important to realize that the type of information stored with the Bitcoin Protocol is independent from the protocol itself. This enables a wide variety of use cases for the Bitcoin Protocol that goes way beyond the implementation of decentralized currency. Indeed the storage of nonvolatile information, as described in the abstract example, is a rather useless and also inefficient application of the protocol. In fact the Bitcoin Protocol makes most sense when we have an ever-growing amount of information, e. g. the history of money transfers between people. In this case we do not have to repackage the same information in every block over and over again, but instead can use them as checkpoints for different versions of the information. There are basically only three aspects in which the different applications which use the protocol vary. One is obviously the type of information that is stored within the blocks. The second one is the process with whom a single node can verify the correctness of a certain block. Lastly, most applications add some kind of incentive to the protocol, which creates an intrinsic motivation for nodes to behave honest and keep the network running.

5.1.1 Bitcoin

Bitcoin makes use of the Bitcoin Protocol to implement a decentralized digital currency. Users can create wallets which are basically a simple Private/Public-Key pair. The most important thing to understand about Bitcoin is, that there is no data structure that resembles a Bitcoin. Instead Bitcoin models the flow of money between the wallets by transaction-objects. A transaction-object contains (among other things) the hashes of the wallets of the participating users, the amount of money transferred between the wallets, and references to older, already valid, transaction-objects which are also called "spending transaction". Spending transactions can be regarded as the source from which the newly created transaction receives its money. The amount of money in a wallet is therefore the sum of all the unspent transaction-objects who contain the hash of the respective wallet as the recipient. Clearly, the transaction-objects are the manifestation of what Bitcoin actually is. Consequently the trustworthiness one can put into such a

system stands and falls with the authenticity of the collection of transaction-objects that one receives from the network. Therefore Bitcoin stores the transaction-objects within the computed blocks. When a node tries to compute a block it not only tries to achieve the desired hash condition but also verifies that the transaction-objects themselves are valid: This includes making sure that no double spending of a previously spent transaction occurs, or that the transaction was actually initiated by the user who submitted the transaction. In the end, the longest chain in the network contains the correct history of transactions that have been performed within in the network. Computing a block in Bitcoin is a fairly advanced task which cannot be carried out by any user who wants to just use Bitcoin as a monetary union. Nodes therefore are service providers for clients who want to place a transaction into the system so they do not have to invest the huge amount computational power to enact a transaction. On average the current Bitcoin network produces one block every ten minutes. This is actually not very fast, but keep in mind that a single block can contain multiple transaction-objects at once (currently a maximum of 2^{512}), so the network is able to serve a huge amount of orders every 10 minutes. All this ultimately begs the question where all the Bitcoins actually come from, if there are only transaction-objects that describe the transfer of existing bitcoins. For this purpose, Bitcoin introduces a special kind of transaction, a so called "generating transaction" which is basically a transaction-object without a source, or existing spending transaction. It literally spills bitcoins from nothing and can therefore only be created by the protocol together with a newly computed block. This principle is often called "mining a Bitcoin" although this terminology is quite misleading as not the actual coin is computed but rather the "right" to shift coins from one source to the other. The idea behind generating transactions is to serve as an incentive for nodes to keep the network running. The amount of coins a generating transaction produces is halved by the protocol every 2016 Blocks (approximately every two months), and will ultimately reach zero when 21 Million Bitcoins are in circulation. The total number of coins is therefore limited, and there will be a time when no new coins can be computed. However Bitcoin maintains another kind of incentive from that time on, in the form of transaction fees that can be gathered by nodes from the clients who place the orders.

5.1.2 Namecoin

Namecoin employs the Bitcoin Protocol to store mappings from domain names to IP addresses in a distributed network. This globally accessible and, given that the protocols assumptions hold, also correct mapping can then be used to perform the same lookup a DNS server would perform, just without having to rely on a trusted party that deployed the DNS server. Namecoin reuses a lot of the codebase of Bitcoin, mainly to simplify the implementation overhead and therefore reuses a lot of concepts introduced in Bitcoin. For example it stores the mappings within the same transaction-objects as Bitcoin. It also has a similar incentive as Bitcoin called Namecoins. Similarly Namecoins are created with the creation of a new block, but can only be spent once to place a new transaction. That means, unlike Bitcoin, there is no continuous flow of Namecoins and the network will eventually run out of them. Namecoin also lets domain mappings expire if they do not occur within the 250 newest blocks. This means that a domain has to be updated at a minimum of every two months to stay within the DNS mapping.

5.1.3 Bitmessage

Bitmessage constitutes probably the most exotic application of the Bitcoin Protocol given in this paper: The goal of Bitmessage is to transfer messages between peers in a secure and encrypted fashion, much like PGP for Mail-services does, expect it is not reliant on a trusted party such as the mail-server or requires some kind of key exchange or a trusted certificate. Instead Bitmessage uses blocks to store the encrypted messages in. This leads to a chain, that basically contains all messages that have ever been send between peers in the network. This means any node in the network can get to any message that has ever been sent, and the Protocol also guarantees that this message is authentic for reasons we have been describing throughout this paper. As all messages can only be decrypted by the node to whom the message was sent, the fact that everyone receives every message does not bear a security risk. Still, every node has to deal with a lot of messages that are of no interest to them. To simplify the access to the messages relevant to a specific node, Bitmessage introduces additional tree-like structures that speed up access times. [13]

6 Future Work

6.1 Usefulness of Proof of Work

Some problems with POW identified in the first publication still exist to this day, namely the uselessness of the computation. While solutions were offered, like the "Bread Pudding Protocol"[4], nothing has yet come close to turning the computation into something really useful. There are quite a few initiatives dedicated to acquiring as much computational power as possible for scientific research and there seems to be no shortage of computationally intensive questions that we would like answered. One can only imagine how useful an infrastructure like the Bitcoin network could be if it was computing something worthwhile in addition to providing its current function.

6.2 Stability of Proof of Work

Hashcash and subsequently Bitcoins POW function are CPU bound. The problem with this was identified by some of the early publications as well, e. g. Hashcash offered another flavor of their MINT function that was non parallelizable [7]. Especially considering the original use case of POW functions, e. g. metering access and preventing abuse, it would seem that fairness towards the regular user should be considered as elemental. The problem with CPU bound algorithms as POW functions is, that CPU speed varies largely across different devices that still might want access to the common resource, consider for example a pda that wants to send an email. Therefore the threshold for the POW function needs to be small enough that even the aforementioned pda can still send an email. But reality suggests that malicious users, which want to send spam are able to acquire a large amount of CPU power for doing so, therefore needing the POW function to have large thresholds to make spam infeasible. This contradiction hinders the use of POW functions and their relevance for their original purpose [14]. While these problems can be solved within the domain of emails by reputation systems [15], another approach would be not to use CPU bound algorithms.
As suggested by Dwork and Naor [16] and refined by Abadi and Burrows [17] memory speed varies much less then CPU speed and therefore can guarantee fairness and ensure the usefulness for POW functions. These thoughts might become important for

the Bitcoin network in the future. Recent introduction of new hardware has shown how much influence one company can have on the Bitcoin network [18, 19]. Since everything depends on the majority of the network being honorable it might become necessary to change the Bitcoin POW algorithm to memory bound since it would require much more resources to take over significant amounts of the network.

6.3 Problems related to Implementation

The Bitcoin Protocol itself is not subject to change dramatically in the future. What however will change and also affect the world as a whole are the different implementations and applications of the Bitcoin Protocol. It is important to realize that the Bitcoin Protocol only delivers an approximate solution to the byzantine fault, which holds as long as the majority of computing power is in trustworthy hands. As of now this might be the case, as the official Bitcoin Network currently outperforms the 500 supercomputers combined by a factor of 8 [20]. This is not only possible because of the sheer number of nodes in the network but also that many of the hardware used by the nodes is specialized for the sole purpose of block computation. In fact, this makes it even harder to estimate the total amount of computational power comprised by the network. Nonetheless there are also flaws in the specific application of the protocol itself. In case of Bitcoin for example, there had been a major bug in the verification of transactions, which allowed users to harvest large amounts of Bitcoins by exploiting an uncatched integer-overflow. The solution of the problem involved an update to the protocol implementation which had to be propagated to all nodes in the network as quick as possible. This lead to a race condition between two chains within the network, one belonging to the nodes stuck at the old version of the protocol, and the other one maintained by the nodes who already adopted to the bugfix. The race luckily turned out in favor of the chain owned by the nodes with the bugfix, but this incident emphasizes that the Protocol itself is not invulnerable, and especially not if bugs exist in the actual implementation.

6.4 Anonymity

Although Bitcoin is often advertised as Anonymous, most implementations do not guarantee anonymity at all. Anonymity is in fact independent from the Bitcoin protocol, and instead bound to the way we store the information inside the blocks in the network, which indicate who talked, send money or interacted with whom. Implementations such as Zerocoin [21] exist, which provide means to completely anonymize the flow of money. To do that, Zerocoin builds upon the idea of Zero Knowledge Proofs which allows someone to proof the ownership of some transaction without conveying that he actually owns this transaction. This is directly related to the concept of plausible deniability, however we cannot go into further detail about this implementation here, as it goes way beyond what this paper tries to cover.

7 Conclusion

We have described the implementation of the Bitcoin Protocol, its concrete applications and the concepts it is based upon. We have shown how the combination of well-known mechanisms created something entirely new. We explained in detail what it is that makes Bitcoin secure and able to compete with other currencies as well as what its up- and downsides are. We are looking forward to further developments with Bitcoin and are very interested to see what challenges arise in the future.

References

[1] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology - CRYPTO'89 Proceedings*. Springer, 1990, pp. 416–427.

[2] R. C. Merkle, "Protocols for public key cryptosystems." in *IEEE Symposium on Security and privacy*, vol. 1109, 1980, pp. 122–134.

[3] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology—CRYPTO'92*. Springer, 1993, pp. 139–147.

[4] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Secure Information Networks*. Springer, 1999, pp. 258–272.

[5] A. Juels and J. G. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks." in *NDSS*, vol. 99, 1999, pp. 151–165.

[6] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996.

[7] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[8] S. Haber and W. S. Stornetta, *How to time-stamp a digital document*. Springer, 1991.

[9] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," in *Sequences II*. Springer, 1993, pp. 329–334.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, vol. 1, p. 2012, 2008.

[11] R. C. Merkle, "Secrecy, authentication, and public key systems." 1979.

[12] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirement," in *the 20th Symposium on Information Theory in the Benelux*. Citeseer, 1999.

[13] J. Warren, "Bitmessage: A peer-to-peer message authentication and delivery system," 2013.

[14] B. Laurie and R. Clayton, "Proof-of-work" proves not to work; version 0.2," in *Workshop on Economics and Information, Security*, 2004.

[15] D. Liu and L. J. Camp, "Proof of work can work." in *WEIS*, 2006.

[16] C. Dwork, A. Goldberg, and M. Naor, "On memory-bound functions for fighting spam," in *Advances in Cryptology-Crypto 2003*. Springer, 2003, pp. 426–444.

[17] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," *ACM Transactions on Internet Technology (TOIT)*, vol. 5, no. 2, pp. 299–327, 2005.

[18] A. Jeffries. (2012) Miner problem: big changes are coming for bitcoin's working class. Accessed: 2013-12-12. [Online]. Available: http://www.theverge.com/2012/11/16/3649784/bitcoin-mining-asics-block-reward-change

[19] E. Rodgers. (2013) 23-year-old releases new chips that 'mine' bitcoins 50 times faster. Accessed: 2013-12-12. [Online]. Available: http://www.theverge.com/2013/2/1/3941768/new-chips-mine-bitcoins-50-times-faster

[20] P. Archer. (2012) Bitcoin network speed 8 times faster than top 500 supercomputers combined. Accessed: 2013-12-12. [Online]. Available: http://thegenesisblock.com/bitcoin-network-8-times-faster-than-top-500-super-computers-combined/

[21] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *IEEE Symposium on Security and Privacy*, 2013.

Introduction to Homomorphic Encryption

Cloud Security Mechanisms Seminar – Summer Term 2013

Hubert Hesse, Christoph Matthies

1 Introduction

When working with confidential data in an untrusted environment some sort of encryption needs to be used. This allows secure transmission as well as storage of the data. However, it also prevents a third party from working with the data unless he receives the decryption key and can operate on and read the plaintext (which might be undesirable). Homomorphic Encryption seeks to tackle this problem: it allows calculations on encrypted data in which the result (after decryption) is the same as if the calculation had been performed on the plaintext.

$$operation(plain) = decrypt(operation'(encrypt(plain)))$$

1.1 Applications

The obvious application for homomorphic encryption is the use in the cloud. Confidential data can be stored in a public cloud[1], operated by an untrusted provider. Third parties could operate on the data, running calculations and analyses without ever gaining any knowledge about the processed data or the meaning of the results.

Another field where concrete solution strategies already exist is electronic voting. Votes can be encrypted and published, preserving election secrecy, while still allowing the encrypted votes to be counted verifiably [1].

There are many more future use cases that could benefit from homomorphic encryption and would allow storing private data in untrusted storage:

Medical records
: Patient data can be stored and analyzed without privacy concerns. Diseases, their treatments and success rates can be analyzed without disclosing them. In the future, DNA markers could be analyzed while preserving the confidentiality of the genome. E-health applications would become feasible with public storage.

Smart metering
: Usage data of electronic appliances in "smart" power grids[2] can be collected and analyzed privately.

Spam filtering
: Currently it is not possible to have a third party scan encrypted (e. g. with GnuPG[3]) E-Mails, without having access to the decryption key. Using homomorphic encryption, costly bayesian spam filtering could be run on undecipherable data while still yielding meaningful spam scoring values.

[1] Cloud resources are made available to the general public, in contrast to a private cloud.
[2] IEEE smartgrids: `http://smartgrid.ieee.org/`
[3] GNU Privacy Guard: `http://www.gnupg.org/`

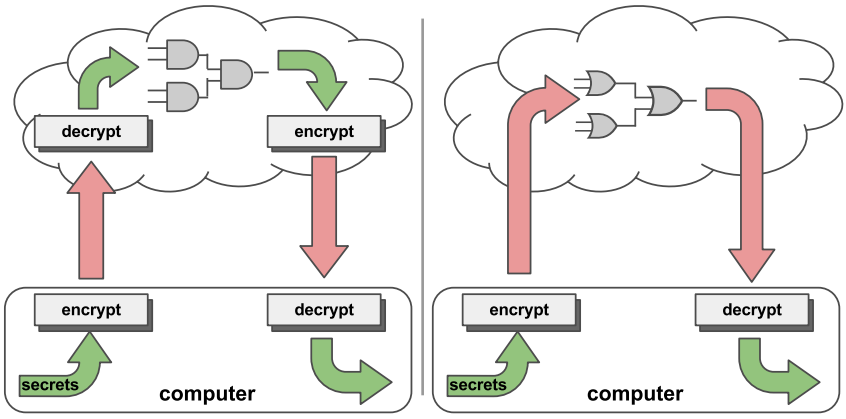

Figure 1: Working with data in the cloud. Traditional encryption (on the left): The cloud provider needs to have access to the key in order to decrypt the data and work with it. On the right: homomorphic encryption where the cloud provider can work on encrypted data.

2 Background

In order to allow such applications as previously listed, the encryption scheme that is used must preserve the properties of the data while it is being computed on. In simple terms this is what is referred to as a homomorphism.

2.1 Group homomorphisms

In general a group homomorphism is a relation between two groups, mapping operands for the the operation defined in the first group to operands for the operation defined in the second one. Imagine the set of all plaintexts P, C the set of all ciphertexts and f the encryption relation. Given the groups $(P, \oplus)$ and $(C, \otimes)$, the relation $f : P \to C$ is a *group homomorphism* for P and C if:

$$\forall a, b \in P : f(a \oplus b) = f(a) \otimes f(b).$$

This definition directly points to our defition of homorphic encryption in Chapter 1:

$$\forall a, b \in P : \underbrace{a \oplus b}_{operation(plain)} = \underbrace{f^{-1}}_{decrypt} (\underbrace{f(a) \otimes f(b)}_{(operation'(encrypt(plain))})$$

Examples of group homomorphisms

In the groups $(\mathbb{R}, +)$ and $(\mathbb{R}^*, \times)$ that we deal with on a daily basis, the exp and log functions are group homomorphisms:

$\exp(x + y) = \exp(x) \times \exp(y)$ $(e^{x+y} = e^x \times e^y)$

and

$\ln(a \times b) = \ln(a) + \ln(b)$.

Both map one operation, here $\times$ and $+$, in the domain of the homomorphism to a completetely different operation in the range.

2.2 RSA - Practical example

When talking about encryption, the RSA cryptosystem is usually mentioned. This is relevant for homomorphic encryption as well, as RSA is homomorphic in regard to the multiplication operation. This is an accidental property as in that RSA was not designed with this result in mind [2]. This property (together with the security level of RSA) can be taken advantage of in a simple cloud computing scenario: Imagine many rectangles that we know the width and height of. We wish to compute their areas, but we do not have the necessary computing power to do so. A third-party service could readily do the necessary multiplication for us, using the rectangle's dimensions, but we do not want to publish this information for, say, privacy or compliance reasons. Using the RSA cryptosystem [3], we can utilize the cloud service's computing power while still maintaining data confidentiality.

2.2.1 Key generation

We generate an RSA keys from two primes (p, q), say $(11, 13)$, such that our base modulus $n = p \times q = 143$. The public key contains a number e which is coprime with Euler's totient of the base modulus n (i. e. $gcd(e, \phi(n)) = 1$);[4] we pick $e = 23$. The private key contains the decryption exponent d, which is the multiplicative inverse of e (modulo $\phi(n)$). That means we need to solve $e \times d \equiv 1 \ mod \ \phi(n)$, which means $d = 47$.[5] As the resulting key pairs must include the base modulus $n = 143$, we arrive at *(23, 143)* for the private key and *(47,143)* for the public key.

2.2.2 Encryption

Imagine our rectangle having dimensions of 7×3. Encryption of a message m in RSA using the public key pair (23, 143) is $encrypt(m) = m^e \mod n$. The encryption of our 7×3 rectangle is thus a 2×126 rectangle:

$encrypt(7) = 7^{23} \mod 143 = 2$. $encrypt(3) = 3^{23} \mod 143 = 126$

This encrypted information is then transmitted over the insecure internet to the third party, non-trusted cloud provider.

2.2.3 Calculation

The untrusted third party can now, without leaking any sensitive data, operate on the data. In our scenario, it calculates the area of a 2×126 rectangle: 252. It must make sure to only use multiplication to calculate the area, as this is the only operation that is a homomorphism in RSA. The cloud provider then return the (still encrypted) result of its calculation back to the client.

2.2.4 Decryption

The client decrypts the returned ciphertext c using his private key pair $(143, 23)$:

$decrypt(c) = c^d \mod n$.

In this scenario the calculation yields:

[4] $\phi(n) = \phi(143) = (p-1) \times (q-1) = 10 \times 12 = 120$ and $gcd(120, 23) = 1$.

[5] Solve $23 \times d + k \times 120 = 1$ with extended euclidean algorithm, $d = 47, k = -9$.

$decrypt(252) = 252^{47} \mod 143 = 21.$
As $21 = 7 \times 3$ is indeed correct, the operation on encrypted data has succeeded.

2.2.5 Proof of homomorphism

It can be easily shown that multiplication in the RSA cryptosystem is a homomorphism. Given the public key pair (e, n), the encryption of a message m is given by $encrypt(m) = m^e \mod n$. Thus, encrypting of the product of two plaintexts a, b is equal to multiplying the ciphertexts:

$$
\begin{aligned}
& encrypt(a \times b) \\
&= (a \times b)^e \mod n \\
&= a^e \times b^e \mod n \\
&= (a^e \mod n) \times (b^e \mod n) \\
&= encrypt(a) \times encrypt(b)
\end{aligned}
$$

3 Implementations

Apart from RSA, there are numerous other encryption schemes that exhibit homomorphic properties. The better known ones are listed in figure 3. Due to the the different subsets of supported operations and mappings of these, not every encryption fits every use case.

System	Plaintext operation	Cipher operation
RSA	$\times$	$\times$
Paillier	$+, -, m \times k, m^k$	$\times, \div, c^k, c \times g^k$
El Gamal	$\times, m \times k, m^k$	$\times, c \times k, c^k$
Goldwasser-Micali	$\oplus$	$\times$
Benaloh	$+, -$	$\times, \div$
Naccache-Stern	$+, -, m \times k$	$\times, \div c^k$
Sander-Young-Yung	$\times$	$+$
Okamoto-Uchiyama	$+, -, m \times k, m + k$	$\times, \div c^k, c + e(k)$
Boneh-Goh-Nisim	$+, -, m \times k, m + k, \times$(once)	$\times, \div, c^k, c \times g^k$, *bilinear pairing*
US 7'995'750[6]	$+$	$+$

Figure 2: Different homomorphic encryption schemes. (m refers to a message, k to a key parameter, c to any constant value.)

3.1 Restrictions

It is worth noting that the simple property of being homomorphic under certain operations, does not make an encryption scheme secure. For example, a simple Ceasar Cipher (e. g. ROT13) would also already allow homomorphic concatenation of ciphertexts, but is in no way secure.

Furthermore, it is vital, that only the operations permitted under the specified homomorphic encryption scheme (see figure 2) are applied to the ciphertexts. Using different operations leads to false results after decryption.

[6] Refers to a US patent concerning "Privacy-preserving substring creation", handed in by SAP AG, published August 9, 2011.

3.1.1 Example

Assume the same scenario introduced in 2.2. However, instead of having the cloud provider calculate the area of our rectangles, we wish to know their circumferences. Using the same RSA encryption of our 7×3 rectangle (circumference $c_p = 2 \times 3 + 2 \times 7 = 20$), we arrive at an encrypted 2×126 rectangle (circumference $c_c = 2 \times 2 + 2 \times 126 = 256$). Decryption of the calculated area (using the private keys of the previous example) yields $256^{47} mod 143 = 42$ which is not the desired result ($20 \neq 42$). Thus, applying the $+$ operation on ciphertexts, which is not homomorph under the RSA cryptosystem, did not translate to an addition on the plaintexts and in fact made the result useless.

3.2 Pollution

Performing operations on encrypted data introduces what is termed "noise" or "pollution", a name for the abstract effect that after a certain number of operations the ciphertext can no longer be decrypted. While addition adds the noise of the operands, multiplication multiplies it. When the noise overshoots a certain threshold (varying with the chosen parameters for the encryption scheme), the decryption fails. Homomorphic encryption schemes that support $+, \times$ with these limitations are called *Somewhat Homomorphic Encryption Schemes*.

3.2.1 Example

A simplified version of the pollution effect can be shown using the previous RSA example. When using large inputs to the operation, such that the result is larger than the RSA-modulus (N, e. g. 143), the decryption will yield a faulty result.

Consider using the same public and private keys as before, but this time, we wish to calculate the area of a square that is 10×15 and will thus result in an area larger than the RSA-modulus. As before, we encrypt the height and width:

$encrypt(height) = 10^{23} \mod 143 \equiv 43$

$encrypt(width) = 15^{23} \mod 143 \equiv 20$

Then the area is calculated in the cipherspace:

$43 \times 20 = 860$.

The decryption step should now produce the correct result of $10 \times 15 = 150$, however, due to the small RSA-modulus, we arrive at a wrong decryption result:

$area = 860^{47} \mod 143 \equiv 7 \neq 150$.

4 Circuit Encryption

Every program that is referentially transparent[7] can be expressed in terms of a digital circuit. This circuit in turn can be represented as a combination of the universal *AND*, *OR* and *NOT* operators (e. g. in a disjunctive normal form[8]). In fact, the *AND* and *XOR* operators suffice to represent any digital circuit.[9]

[7] An expression is considered referentially transparent if it can be replaced with its computed value without changing behavior, e. g. a call to *today()*, returning today's timestamp, would not be referentially transparent [4].

[8] Normalization of a logical formula as a disjunction of conjunctive clauses.

[9] Any function can be written as a combination of them as well: $x \oplus True = \neg(X)$ and $\neg(\neg x \wedge \neg y) = x \vee y$.

Brenner et. al [5] introduced a scheme we can use to homomorphically encrypt such boolean algebra representations of programs. The main idea is to encrypt a plaintext bit by encoding it as an integer and then encrypting the resulting integers.

1. Map 0-bits to arbitrary even integers $< p$. Add random multiple of secret p.
2. Map 1-bits to arbitrary odd integers $< p$. Add random multiple of secret p.
3. Map algebraic $\oplus$ to integer $+$.
4. Map algebraic $\wedge$ to integer $\times$.
5. Define $a \circ b = (a + b) + (a \times b)$ ($a \vee b = (a \wedge b) \wedge (a \oplus b)$ in hormorphic space).

We can encrypt the corresponding integers by adding a random multiple of a secret prime. Without the correct modulos p it is impossible to decide whether an integer represents a 1 or a 0 bit in plaintext.

This allows us to reproduce boolean operations, thus any circuit, by integer multiplication and addition. Note that this scheme fails if the results of the integer operations growths larger than the modulus p (see section 3.2).

4.1 Example: Encrypting a Single Bit Adder

In this section we give a simple working example of an encrypted single bit adder.

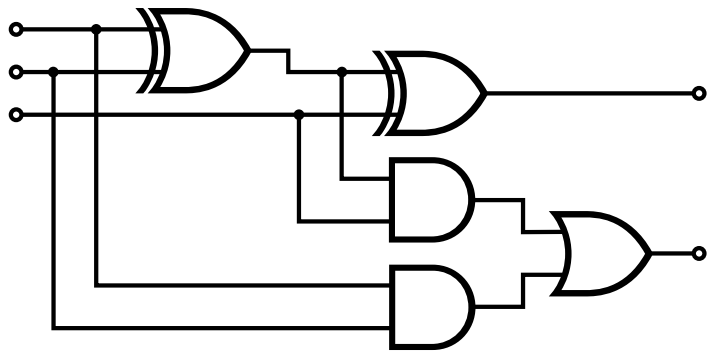

Figure 3: Logical circuit of a single bit adder.

The boolean term to describe the circuit can be trivially read from figure 3.
$S = ((A \oplus B) \oplus C)$
$C_{out} = (A \wedge B) \vee (A \oplus B) \wedge C_{in})$

With the mapping scheme of Brenner et. al (see section 4), the boolean terms translate to:
$S = ((A + B) + C)$
$C_{out} = (A \times B) \circ (A + B) \times C_{in})$

Let's add $A = 1$ and $B = 0$ with a carry in bit of 1:
$S = ((1 \oplus 0) \oplus 1) = \mathit{0}$
$C_{out} = (1 \wedge 0) \vee (1 \oplus 0) \wedge 1) = 0 \vee 1 = \mathit{1}$

Now we simulate this operation in homorphic space according to the hormophism declared in the previous section. Therefore we choose a secret p = 23, translate A to 3 (an

odd integer), B to 4 (an even integer) and carry in bit (C_{in}) to 7 (another odd integer). Further we add random multiples of 23:

$A = 3 + 2 * 23 = 49;$
$B = 4 + 5 * 23 = 119;$
$C_{in} = 7 + 3 * 23 = 76$

Note how this prevents a third party from distinguishing odd from even plaintext bits.

$S = ((49 + 119) + 76) = 244 \; mod \; 23 \equiv 14 \mathrel{\widehat{=}} 0$
$C_{out} = (49 \times 119) \circ (49 + 199) \times 76)$
$= 5831 \circ 15173$
$= (5831 + 15173) + (5831 \times 15173)$ (with defintion of$\circ$)
$= 21004 + 88473763 = 88494767 mod \; 23 \equiv 13 \mathrel{\widehat{=}} 1$

We are able to encrypt a single bit adder such that neither the input or the output values are revealed, while the results are preserved. Note that we worked with a simplified representation model with only a small random multiple of p.

4.2 Encrypted Memory Access

Although arbitrary cicuits can de encrypted, memory access patterns of algorithms could be leaked and a third party could deduce hints to the nature of the program or possible attack vectors on it. For example, configuration paramters could be read from the same spot in memory at the start of every program run. Since any circuit can be encrypted, we can also build a static memory scrambler in homorphic space. Brenner et. al [5] propose a solution with 4 memory bits and 2 address bits.

The basic idea is to mix up the usual memory access, e. g. address 01 points to memory cell 2 instead of cell 1, address 10 to cell 1 and so on. This is achieved by accessing the memory bit through a series of AND gates. The address bits are wired into these AND bits with NOT[10] gates in a way that the AND gate for a memory cell only switches and reveals the value of the memory bit on the correct assignment of the address bit for that memory cell. All the AND gates are connected to a connecting OR gate that returns the value of the access memory bit. With this method it is possible to access encrypted memory with an "encrypted" memory adress such that neither memory address accessed or content are revealed to a third party, that does not know the address wiring for this program. The concrete address wiring can be changed at compile time, e. g. for every binary of the homorphically encrypted program.

5 Fully homomorphic encryption

In the previous sections we presented homomorphisms that mapped certain operations to others. Therefore, specific homomorphisms have niche applications where this operation is required. Ideally one would find an encryption which would allow arbitrary operations on the ciphertext. Such schemes are called *fully homomorphic encryptions* (FHE). As already discussed in section 4 every circuit (and therefore every program) can be expressed as a boolean algebra term. Thus, any homomorphic cryptosystem that supports binary *AND* and *XOR* — neatly represented as multiplication and addition on multiple bits — is fully homomorphic.

[10] NOT Gates are implemented with value XOR 1.

5.1 History

Only one year after Rivest, Shamir und Adleman presented their RSA system in 1977 [3], Rivest, Adleman and Dertouzos proposed the concept of fully homomorphic encryptions (they called these *privacy homomorphisms* [6]).

For the next thirty years, no significant headway was made in the field of fully homomorphic encryption. However, in 1999, Paillier presented his scheme [7], which already allowed many important operations on ciphertexts (see Table 3). Boneh, Goh and Nissim proposed a solution in 2005 [8], which did allow both addition and multiplication, coming closer to the goal of FHE. However, only a single multiplication could be performed before the ciphertext became undecipherable.

The breakthrough came in 2009 when Craig Gentry, an IBM researcher, proposed a new (and working) approach in his dissertation [9].

5.2 Gentry's FHE scheme

Gentry's idea was to use a *somewhat homomorphic encryption* (SWHE) as a stepping stone to a FHE using a process he called *bootstrapping*. SWHEs can only allow a certain number of operations before the noise is too large, rendering the cipher unusable.

This is comparable to an error correcting code: If the error or noise is small, the recipient can use the knowledge of the code to remove the noise; if it is too large, not even the error correcting code can help with recovery. This effect can be counteracted by decrypting the data whenever the noise reaches critical levels and encrypting it again afterwards — a process commonly referred to as reencryption. It ideally reduces the noise to former levels. This would of course requires knowledge of the secret key which contradicts the goal of computing in an untrusted environment.

Gentry managed to solve this problem by allowing the encryption scheme (which can evaluate arbitrary circuits) to recursively evaluate its own decryption circuit — the aforementioned *bootstrapping*.[11] Running the decryption on the ciphertext with an encryption of the secret key produces a reencryption of that cipher with reset noise. One condition that needs to be met is that the decrypt circuit itself must be simple enough to not exceed the noise threshold after which the message is scrambled. When Gentry first formulated his scheme, he found that this condition was not met. The remedy was "squashing" the decrypt circuit at the cost of a longer key size.[12] With the reencryption step being repeatable as needed, the scheme could evaluate any circuit of finite depth and the system became fully homomorphic.

The big drawback to Gentry's scheme was performance, both encryption and reencryption took a long time in practice. Encrypting a single bit took about 19 seconds. However, since Gentry had based his encryption on the hard problem of *ideal lattices*[13] This has implications for future security of the scheme. Whereas the problem of integer factorization (used in e. g. RSA) is *quantum-broken*, meaning it can be solved in polynomial time on a quantum computer using Shor's algorithm [12], the problem of lattices seems NP-hard (still, that is).

[11] This concept is possibly hard to grasp at first. An analogy could be the human genome, encoding in its DNA the proteins needed to create more DNA and replicate. In essence the information needed is recursively contained in the element itself.

[12] Squashing describes the "procedure [of] reducing the degree of the decryption polynomial. This is done by adding to the public key an additional hint about the secret key." [10].

[13] A special class of lattices (essentially, mathematical groups) with some additional algebraic structure, often used in cryptography [11].

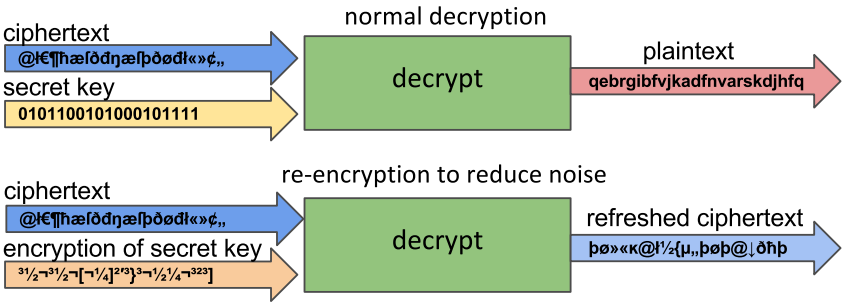

Figure 4: Traditional encryption (top) vs. reencryption in Gentry's FHE scheme (bottom).

5.3 Related work

In the years following Gentry's publication, many variations and improvements of the scheme have been proposed. Most of these approaches share the same approach that Gentry took, promoting a somewhat homomorphic encryption to a fully homomorphic one using bootstrapping, but offer up to four orders of magnitude of performance enhancements.

Van Dijk, Gentry and Halevi managed in 2010 to simplify Gentry's concept by using addition and multiplication over the integers rather than working with ideal lattices [13]. In the same year, Smart and Vercauteren tackled the problem of the large key sizes, managing to reduce their size [14]. At the Eurocrypt '10 conference, Gentry and Halevi were able to present a working implementation of FHE [15]. They had originally planned using IBM's Blue-Gene supercomputer, but ended up only needing a strong workstation. However, for large inputs (13,000,000-bit integers) the secret key still needed to be 2.3 GByte in size and the reencryption step took 30 minutes.

Building on this work, Coron et al. managed in 2011 to reduce the public key size from Gentry's $\tilde{O}(\lambda^{10})$ to $\tilde{O}(\lambda^{7})$[14] while retaining roughly the same level of efficiency [16]. Smart et al. showed in the same year, that SIMD operations[15] could be used in conjunction with FHE, allowing parallel encryptions (especially also parallel reencryption steps) [17].

In an effort to attack the underlying problem of having to reencrypt the cipher periodically, Brakerski, Gentry et al. proposed in 2012 an approach using *leveled* fully homomorphic encryption schemes, which work without the bootstrapping procedure, again increasing performance [18].

The quest to find a practical FHE scheme is not merely of academic nature. In 2011 DARPA and its intelligence counterpart, IARPA both launched programs (PROCEED[16] and SPAR[17], respectively) with explicit goals to make homomorphic encryption practical. The PROCEED program, with its $20 million budget, aims to reduce the computing time by a factor of 10 million and has disclosed that a two orders-of-magnitude speed increase of FHE has been achieved within one year.

[14] $\tilde{O}$ is a variant of the big-O notation ignoring logarithmic factors.

[15] **S**ingle **i**nstruction, **m**ultiple **d**ata operations: performing the same operation on multiple data points simultaneously.

[16] `http://www.darpa.mil/Our_Work/I2O/Programs/PROgramming_Computation_on_EncryptEd_Data_(PROCEED).aspx` (accessed 01.07.2013).

[17] `http://www.iarpa.gov/Programs/sso/SPAR/spar.html` (accessed 02.07.2013).

5.4 Criticism

It has been criticised that "it will be years before a sufficient number of cryptographers examine the algorithm that we can have any confidence that the scheme is secure"[18]. We believe this to be a valid argument as up to this point research on this topic has been mainly performed by researchers around Gentry et al.

5.4.1 HELib

There are, however, implementations of homomorphic encryption, such as HELib[19] by Shai Halevi, that are open-source, GPL-licensed and encourage participation. HELib already includes many of the proposed perfomance optimizations of homorphic encryption schemes. however, bootstrapping is not implemented yet. Using this model, the security of homomorphic encryption and its implementations can be advanced at a faster pace.

6 Conclusion

Today, there are already some very interesting implementations of somewhat homomorphic encryption, for example CryptDB, an encrypted database that allows executing SQL queries over encrypted data using homomorphic schemes as part of a collection of SQL-aware encryption algotithms. Implementations of fully homomorphic encryption exist but are not production-ready just yet; considering that is was only in 2009, that Gentry was able to show that fully homomorphic encryption is possible at all, the progress seems impressive though.

Gentry argues that FHE might lead to more privacy and data confidentiality in general as it is not necessary to decrypt data in order to work with it, removing a barrier keeping people from doing so in the first place. Even if this is not the case FHE seems like a very good solution to a lot of issues concerning data security (especially online) today.

The major challenge for widespread adoption is still performance. However, with funding from government agencies and more universities outside the orbit of Gentry et al. setting up research projects dealing with homomorphic encryption[20], the path seems prepared for further developments in this area. Halevi predicted in a talk in 2012[21] that homomorphic encryption "should be usable in niche applications within a year or two".

References

[1] K. Peng, R. Aditya, C. Boyd, E. Dawson, and B. Lee, "Multiplicative homomorphic e-voting," *Progress in Cryptology-INDOCRYPT 2004*, pp. 1403–1418, 2005.

[2] L. A. Ronald L. Rivest, Adi Shamir, "Cryptographic communications system and method," Patent US 4 405 829, 09 20, 1983.

[18] Homomorphic Encryption Breakthrough, Schneier on Security, Bruce Schneier.

[19] HELib: `https://github.com/shaih/HElib`.

[20] HomER (HOMomorphic Encryption Realization) project of the Karlsruhe Institute of Technology, `http://www.iks.kit.edu/project-homer` (accessed 01.07.2013).

[21] Recent Advances in Homomorphic Encryption, presentation by Shai Halevi, IBM Research, 13.02.2012, `http://nms.csail.mit.edu/sys-security/FHE.pptx`.

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

[4] H. Søndergaard and P. Sestoft, "Referential transparency, definiteness and unfoldability," *Acta Informatica*, vol. 27, no. 6, pp. 505–517, 1990.

[5] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith, "Secret program execution in the cloud applying homomorphic encryption," in *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*. IEEE, 2011, pp. 114–119.

[6] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT'99*. Springer, 1999, pp. 223–238.

[8] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," *Theory of Cryptography*, pp. 325–341, 2005.

[9] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[10] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, ser. EUROCRYPT'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 129–148.

[11] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," *Public Key Cryptography–PKC 2008*, pp. 162–179, 2008.

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

[13] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," *Advances in Cryptology–EUROCRYPT 2010*, pp. 24–43, 2010.

[14] N. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," *Public Key Cryptography–PKC 2010*, pp. 420–443, 2010.

[15] C. Gentry and S. Halevi, "A working implementation of fully homomorphic encryption," *Eurocrypt 2010 rump session*, 2010.

[16] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," *Advances in Cryptology–CRYPTO 2011*, pp. 487–504, 2011.

[17] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, Codes and Cryptography*, pp. 1–25, 2011.

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *Innovations in Theoretical Computer Science, ITCS*, pp. 309–325, 2012.

Differential Privacy

Cloud Security Mechanisms Seminar – Summer Term 2013

Toni Mattis

1 Introduction

Whenever databases are compiled there is a natural interest in both *individual contributions* and *aggregated knowledge* about the data. Given e. g. climate data or records of astronomical events there are obviously little to no negative implications satisfying both needs. However, many databases are the result of collecting *sensitive* personal or business-related records, e. g. censuses, medical records or phone tracking data. In this case, the public release of any individual's contribution, e. g. income, diseases or locations can have a severe impact on its *privacy* while high-level knowledge can be life-saving, e. g. identifying correlations between risk factors and diseases. Hence, a lot of research has been done on how to balance the need for knowledge from aggregated data against the privacy needs of the contributors.

1.1 Problem statement

Consider a database containing the annual sales of some companies maintained by a census office. The sales are considered sensitive information which should not be revealed to any member of the public.

Company	Municipality	Sales (USD)
Roundcat Inc.	Bridgeshire	100,000
Sumtexon Inc.	Bridgeshire	200,000
Quotedax Inc.	Bridgeshire	3,000,000
Triotech Inc.	Fairwalk	4,000,000
Warelane Inc.	Fairwalk	1,000,000

Figure 1: Example Census Database

The census office may receive two requests: The first one queries for the total sales of all companies in the respecting region (8.3 Mio), the second one is interested in the total sales in Bridgeshire (3.3 Mio). In an attack scenario, both queries could be issued by a representative of Triotech Inc. who knows the sales of its own company. Computing the sales of the competitor Warelane Inc. becomes trivial then: $8,300,000 - 3,300,000 - 4,000,000 = 1,000,000$.

The example shows the most relevant problem in privacy-preserving data mining and publishing, the **inference problem**: Even though individual information is never disclosed directly, combining multiple answers and background knowledge allows to infer non-obvious sensitive information.

Besides the direct **inference** of the sensitive attribute there are related **threats** to an individual's privacy:

- **Linkage attack:** A non-anonymous data source which overlaps largely with the (anonymized) sensitive database is available. The overlapping attributes may give a strong evidence which anonymous record belongs to which individual in the non-anonymous database (e. g. identifying a pair of records from a sensitive clinical study and a public voter's register which coincide in city, age and occupation could lead to the voter in question being linked to his diagnosis).

- **Probabilistic attack:** An attacker may get to know with which probability an individual has a certain attribute. Privacy is compromised if that probability differs significantly from what an attacker might believe before its attack (e. g. he might infer from clinical records that a person might have HIV with higher probability as the average population).

1.2 Goals

Most privacy preserving methods (although mostly not explicitly stated in literature) focus on the following goals to mitigate inference, linkage and probabilistic threads:

1. **Uncertainty:** Making any reasoning about individual contributors too uncertain, even in the presence of (limited) background knowledge or additional databases, e. g. by making many individual contributors indistinguishable. This includes reasoning about exact values as well as reasoning about possible probability distributions.

2. **Plausible Deniability:** Making it impossible for an attacker to prove an individual's contribution to an aggregate result to a third party, e. g. using randomization so that potentially revealing results might also be coincidence.

3. **Utility:** Sacrificing not too much accuracy concerning aggregations and data mining operators, e. g. sums and correlation measures. This can either be inferred from a mathematical model (as in ϵ-*differential privacy*) or achieved by minimizing some error measure.

Moreover, some more recent methods address additional challenges which have proven to be rather difficult:

1. **Updatability:** Maintaining privacy even when multiple versions of a database are accessible as proposed in *m-invariance* [1]. Common methods actually assume immutability of the database and therefore ignore privacy threats emerging from incremental updates.

2. **Personalization:** Accommodating to individual privacy levels and preferences, e. g. increasing the data accuracy of contributors with low privacy requirements while protecting those with stricter privacy preferences, as in [2].

3. **Distributedness:** Allowing multiple data sources which do not trust each other to collaboratively compute an aggregate result without revealing any individual contributions. This topic is mainly subject to research in *secure multi-party computation* (SMC).

1.3 Scenarios

Basically, there are two fundamental scenarios which lead to different solutions approaching the goals above: An interactive scenario proposing a central trusted entity and a non-interactive scenario making data accessible without being able to exert further control over the published information.

1.3.1 Interactive scenario

In this scenario, the raw database (including each individual record) is maintained by a central, trusted entity. Members of the public can issue queries against an interface. Those queries may be arbitrarily expressive and complex, so the trusted entity needs to decide on how to answer each query without compromising any individual contribution. Typical examples are online census databases, e. g. from the decennial U.S. Censuses or the 2011 census in Germany.

1.3.2 Non-interactive scenario

The most common non-interactive scenario is data publishing. Once distributed in public, there is no chance to stop further analysis of any database. The central problem is the right choice of an anonymization method anticipating even future attack vectors. Typical examples are customer data being sold to market research companies or medical records published by hospitals. A non-interactive release does not necessarily be a classical database, it can be just an aggregated result for which the data has been collected (e. g. a histogram stating how a population answered a certain question).

1.4 Protecting the aggregated result

If an individual contribution could have a severe impact on the aggregated result, then observing that impact might reveal this particular individual's contribution. Hence, the mechanics of some privacy mechanisms also work in the opposite direction: They protect aggregated results from being significantly manipulated by individual contributors. Especially ϵ-differential privacy can be used to build both a privacy-preserving data-mining mechanism and manipulation-resistant mechanisms. A theoretic scenario would be an online auction (e. g. for selling stock) where bidding agents have no benefit from bidding strategically [3].

1.5 Solution strategies

There are multiple ways of trading uncertainty and plausible deniability against utility. The most common mechanisms choose one of the following strategies:

- **Reject** queries leading to a privacy risk
- **Obscure the data** by making individuals indistinguishable (*microaggregations*, *k-anonymity*, *l-diversity*, *t-closeness*) or noisy
- **Obscure the query processing** by randomly dropping and adding individuals from and to each aggregation (*random sampling*)
- **Obscure the result** by adding random noise to the output (*ϵ-differential privacy*)

1.6 Structure of this report

The following section is concerned with some fundamental vocabulary and concepts. The history section will explain the origins of private data mining and publishing, introduce some historic algorithms and proceed to modern approaches, such as k-anonymity and *differential privacy*, where current research lines split. The mechanism details section will explain the state-of-the-art mechanics of l-diversity, t-closeness and ϵ-differential privacy. In future work we refer to several open problems concerning both lines of research and the conclusion summarizes current and future roles of both lines of research.

2 Foundations

We hereby define some terms which will occur in this report.

2.1 Aggregations

An *aggregation* or *query* is considered any function which takes the full database as input and delivers any output, the *aggregated result*. We mostly imply that the output is numeric. Common examples used as aggregations are *count* queries, which ask how many records in the database satisfy some non-trivial predicate, *subset-sum* or simply *sum* queries which sum up values if the corresponding record satisfies some given predicate, *average*, *standard deviation* or *co-variance* and other statistical measures. The term *query* refers to an aggregation function given by a user in an interactive scenario.

2.2 Mechanisms

When used in a mathematical context, a *mechanism* is a generalization of a *function* in such a way that it yields a probability distribution over the possible results. Each function is a mechanism that outputs probability 1 for the function value and 0 otherwise.

2.3 Quasi-Identifiers in a relational model

All methods presented here assume something similar to a relational database consisting of one or more relations (tables). When personal data is collected, usually some unique key is inherently associated with the data, i. e. the full name of the person, which has to be removed or falsified first.

However, even if each record is anonymized at its own, there may still be a combination of attributes which associate a record with an entry in a different database. e. g. date of birth, ZIP code and occupation of a medical record owner could be linked to a public voters list containing those three attributes in addition to name, address and party affiliation thus linking the identity of the record owner with its diagnosis. Such a combination of attributes which is likely to enable *linkage* to external databases is called ***quasi-identifier* (QID)** [4].

3 History

3.1 U.S. Census

In the 1960s, the decennial U.S. Censuses have raised scientific concerns over how requests to the census offices are handled. The guidelines intended for human decision-

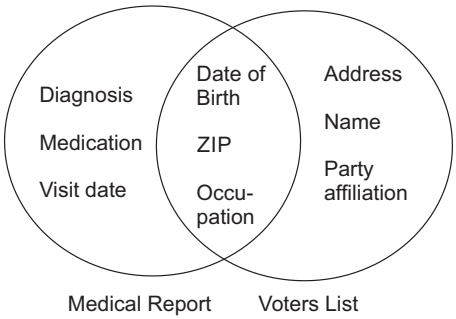

Figure 2: Quasi-identifiers: attributes linking anonymous and non-anonymous data sets

making were eventually generalized to be applied by the emerging relational database management systems. These first privacy mechanisms were designed to either give exact results or reject the request based on the following criteria [5]:

- **Minimum Query Set:** Queries covering too few individual records were rejected.
- **Minimum/Maximum Overlap:** Queries overlapping previous queries in too few or too many individual records were rejected. Considering the introductory example, the census office should not answer one of the two queries whose query sets differ in only two company records.
- **Auditing:** Keep track of all information published and reject queries which would lead to a system of equations being solvable for any single individual or small group.

Giving exact answers comes with the price of a very resource-intensive auditing process and eventually leads to the rejection of more and more queries. A dilemma occurs when queries have different priorities but arrive at random times. The census office may then be well-advised to preemptively defer some queries (e. g. those concerned with random subsets of cities or regions) to be able to answer later queries which are of higher interest for the public (e. g. those about whole cities or regions). Hence, balancing short answer times against utilitaristic principles is delicate [6].

3.2 Query Set Approximation

The drawbacks inherently associated with exact answers were eliminated by making the sets over which aggregations are computed inaccurate. This way an adversary does not know over which individual records the final result was actually computed while still getting a useful approximate answer.

Two different but representative methods are the *random sampling* based query set approximation and *micro-aggregations* [5].

3.2.1 Random Sampling

The random sampling method [7] compares a fixed-length hash $H(q) \in \{0,1\}^n$ of the query with a (precomputed) hash $h(d_i) \in \{0,1\}^n$ of each database entry d_i. Equal hashes

Name	Age	Income p.a.
Person A	30	$40.000
Person B	35	$60.000
Person C	40	$30.000
Person D	45	$80.000

Name	Age	Income p.a.
Person AB	30 - 39	$50.000
Person CD	40 - 49	$55.000

Figure 3: Raw and microaggregated data example

result in an entry being retracted from the query set. The probability of an n-bit entry hash colliding with the query hash is 2^{-n}, so for each query considering k elements, statistically $k*2^{-n}$ elements are removed. Both hash functions h and H must be kept secret.

Later versions of this algorithm also accounted for the tendency to *underestimate* sums and counts by adding approximately as many pseudo-randomly selected elements as removed from the initial query set.

Provided that the query hash is computed on some normalized form of the query independently of syntactic details or equivalent formulations, a property of this pseudo-random algorithm is the fact that asking the same query multiple times will always generate the same result. Nevertheless, an attack is possible if the attacker splits its original query into multiple queries that can be combined to the final result. e. g. in order to obtain a full sum over incomes, he may issue two queries summing up the income of male and female individuals and do the final addition himself. Then he could query for the sum over married and unmarried individuals, getting a second equivalent answer, and so on, until the noise cancels out.

3.2.2 Microaggregation/Partitioning

The method of *microaggregations* [5] first partitions the data and then precomputes statistical measures (e. g. count, average and standard deviation of the sensitive attributes) for each partition. When computing an aggregated query result each partition is either fully included or not at all. Partitions therefore serve as *average individuals* approximating the exact value distribution of the underlying individuals while concealing their individual contributions.

Given the minimum partition size as privacy constraint, finding a partitioning which minimizes the information loss is considered computationally hard. Practical solutions choose a partitioning which addresses the most common query criterion, e. g. demographic data (censuses) should rather be partitioned by neighborhoods than by age as querying the average age of people living in Manhattan seems more reasonable than querying the average location of people aged 30.

Provided that partitions are large enough that no further query restriction needs to be applied, the aggregated data can be released to the public. This makes this approach suitable for interactive as well as non-interactive scenarios. Due to its simplicity and the quite acceptable privacy level achieved it is still attractive nowadays, although it has neither a mathematical privacy model allowing to prove that every individual is sufficiently protected nor a very high utility for fine-grained queries.

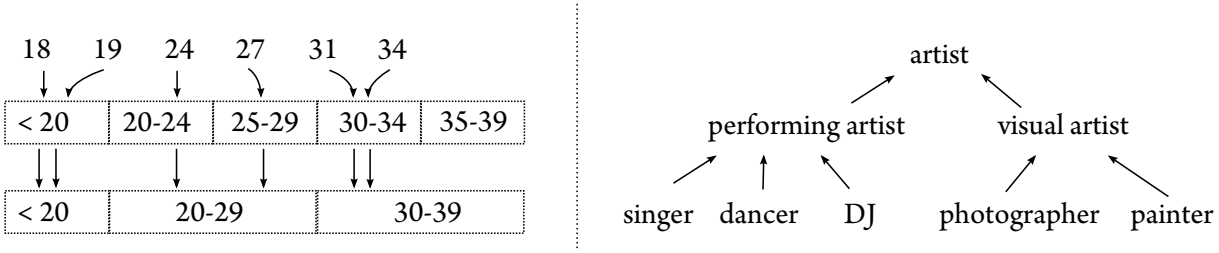

Figure 4: Generalization by intervals vs. generalization by taxonomies

A major drawback of *microaggregations* is the falsification of statistical measures which have not been anticipated. e. g. if each partition's mean of some attributes is given then the correlation between those attributes is essentially destroyed on partition-level.

3.2.3 Drawbacks

Both methods presented above are unable to accept incremental updates to the database. An attacker observing a change in a sensitive average value (e. g. average income of a neighborhood) can deduce the contribution of the updated single individual (e. g. the new neighbor moved in) or small group.

3.3 Anonymization by generalization

Generalization-based approaches, also referred to as *syntactic* approaches rely on the assumption that the sensitive values can always be published in an unmodified manner if the individuals itself cannot be identified with high confidence. Unmodified sensitive values are an important requirement for pharmaceutical studies where adverse effects need to be investigated with maximum accuracy, so artificial sensitive values from microaggregations or random sampling induce intolerable risks for patients.

In order to enforce strict upper bounds on confidence, groups of attributes which could isolate a single individual (QID) have to be identified and their values generalized so that multiple individuals effectively become indistinguishable. Those *equivalence classes* of indistinguishable individual records will be referred to as *QID groups* from now on.

3.3.1 Generalization mechanics

Consider a medical database containing ZIP code, occupation, age and diagnosis of several patients. Diagnosis is considered a sensitive attribute while the combination of ZIP, occupation and age could effectively identify some individuals if the attacker has the corresponding background knowledge. The following generalizations can be made on the QID attributes [4]:

- **Interval generalization:** Continuous or file-grained discrete values (e. g. age given in years) can be discretized into larger intervals (e. g. decades).
- **Taxonomic generalization:** Given a taxonomy over a categorical attribute, the sensitive values can be (repeatedly) replaced by parent nodes inside the taxonomy. e. g. the occupation attribute values *dancer* and *singer* can be generalized to *performing artist* or even *artist*.

- **Suppression:** If a value has no meaningful generalization but needs to be generalized (e. g. the root of the taxonomy tree, gender, URLs, images, ...), the value can be completely retracted or replaced by a meaningless placeholder like "any".

3.3.2 k-anonymity

k-anonymity [8] is a privacy constraint which requires the data to be generalized in such a way that no QID group contains less than k individual records. The concept effectively hides any individual among $k-1$ other individuals.

Although k individual records are indistinguishable, *k-anonymity* is a very weak privacy constraint. An attacker targeting a single individual can obtain all sensitive attributes of the group and be certain, that the victim cannot have any other value. In the worst case scenario the attacker is left with a single sensitive value shared by all group members without leaving any plausible deniability.

An attacker also learns the distribution of the sensitive attributes from the group, which can be considered a privacy risk if it significantly differs from the overall distribution. Considering e. g. employers rejecting or dismissing employees as they learn that the employee in question could have certain diseases with much higher probability than usual shows this vulnerability of plain k-anonymity.

A third weakness of k-anonymity is its assumption that each individual is represented by a single record. A record owner having many records contributed to the database may be underprotected by k-anonymity.

3.3.3 (X, Y)-anonymity

(X, Y)-anonymity [4] is an important generalization of k-anonymity. X and Y represent attributes (or groups of attributes) in the database. When *(X, Y)-anonymity* is given, for each unique value in attribute X there are at least k unique attributes in Y. The basic k-Anonymity constraint presented above can be seen as an instance of *(X, Y)-anonymity* where X is the QID and Y is a key or ID unique to each record.

If Y refers to an ID/key for the record owner (e. g. its full name) instead of the record ID, this effectively ensures that at least k record owners are contained within a QID group which prevents any individual from being accidentally underprotected by k-anonymity.

3.3.4 l-diversity

In order to mitigate the problem that an attacker may rule out too many sensitive attributes by identifying the victims QID group, the *l-diversity* [4] constraint ensures that at least l different values are left for each sensitive attribute. Therefore l-diversity facilitates that individuals may plausibly deny $l-1$ values attributed to them. l-diversity automatically fulfills l-anonymity as there must be at least l entries in each QID group. If there is only one sensitive attribute, l-diversity can be modeled as an instance of (X, Y)-anonymity with X being the QID and Y being the sensitive attribute.

3.3.5 t-closeness

The *t-closeness* [4] constraint is even stricter than l-diversity as it forces the value distribution inside each QID group to differ by at most t from the overall distribution. Depending on whether the sensitive attribute has a numeric or categorical character, t can be computed using *Earth-Mover's Distance*, *Kullback–Leibler (KL-)divergence* or any other common method measuring the distance of two distributions.

3.4 Differential privacy

Although they are very useful for research on unchanged sensitive values, generalization based approaches suffer from the drawback that their privacy guarantees are not considered strong enough.

Differential privacy therefore takes a stricter approach concerning provable guarantees. It does not directly protect the original data but obfuscates every processing of the data (e. g. statistics) by adding a carefully chosen amount of noise to the output [9].

Noise camouflages individual contributions. It can be chosen proportionally to the amount of change a single individual may have on the aggregated result, so it is uncertain whether an aggregated result actually involves some specific individual's contribution or has been altered by coin-flipping [10]. This guarantees plausible deniability to each single individual with some adjustable *plausibility*.

Differential privacy also guarantees that the information an arbitrarily well informed attacker gains after seeing the result is nearly independent of whether an individual contributed to the result or not. Hence, it defends against attacks with unlimited background knowledge [11].

The mechanisms presented in the context of differential privacy primarily focus on non-interactive scenarios, e. g. some trusted entity releasing aggregated or otherwise processed knowledge about candidates taking a survey, but it can be easily adapted to interactive scenarios if the number of possible queries to the system is limited.

In 2012, five years after the formal concept of differential privacy emerged, Ghosh, Roughgarden and Sundararajan [11] proposed the first universal *utility model* for count queries and presented an improvement over the earlier privacy-utility-tradeoff for a very restricted type of queries.

3.5 Bringing everything back together

One big issue with k-anonymity based approaches (and also microaggregations) is the algorithm used [12]. The actual partitioning of the data in QID groups (or average individuals) can, if the algorithm is understood by an attacker, reveal a considerably large amount of original data to the attacker provided he has sufficient background knowledge. This is also due to the fact that a single individual, if present, could change the complete partitioning of the data and therefore have a massive impact on the results, which is what differential privacy strictly confines.

In response to the weaknesses of k-anonymity 2011, Li, Qardaji and Su proposed the usage of ϵ-differential privacy in the context of k-anonymity which they called ϵ-safe k-anonymization [13].

3.6 Summary of recent developments

On the one hand, k-anonymity, (X, Y)-anonymity, l-diversity and t-closeness are former and current state-of-the-art models for anonymizing the basic data without anticipating the type of aggregation. They emerged from the need for true sensitive values, especially in the life-critical field of medical research and focus on preventing *linkage attacks* by making groups of individuals virtually indistinguishable.

On the other hand, differential privacy is a concept concerned with a result computed from the data, so it is closer to the original roots of privacy research assuming some central entity doing the actual aggregation rather than just anonymizing raw data. However, it does neither change the query set, like random sampling or k-anonymity, nor rely on microaggregated data and thus improves the utility of the result drastically over historic approaches. It effectively defends against *probabilistic attacks*.

4 Mechanism Details

The following section explains the two current branches in the line of privacy research: l-diversity and t-closeness as state-of-the-art mechanisms for privacy-preserving data publishing and differential privacy as emerging mechanism for privacy-preserving data mining.

4.1 l-diversity and t-closeness in practice

4.1.1 Entropy l-diversity

From an information-theoretic perspective, different sensitive values contribute a different amount of information to a QID group. The entropy of QID group G is defined as $H(G) = -\sum_i P(s_i) log_2(P(s_i))$, where s_i are the different sensitive values in G, $P(s_i)$ is the probability of s_i occurring and $H(G)$ gives the expected number of bits for each sensitive value in the QID group. The number of bits necessary to represent l equally probable values is $log_2(l)$.

This leads to a state-of-the-art definition of *l-diversity* which is satisfied if for each QID group $H(G) \geq log_2(l)$ holds.[4] In other words, the expected information content of a sensitive value must be at least the baseline information content when all sensitive values were equally probable.

Although entropy l-diversity makes the distribution of sensitive values less revealing in the eye of an attacker, this method still lacks an exact control mechanism for the *probability density function* (PDF) an attacker may deduce from prior knowledge combined with the data. This problem will be addressed by both t-closeness and differential privacy.

4.1.2 Semantic l-diversity

It is important to know that l-diversity can be ineffective when the l sensitive values are semantically close. Consider a 3-diverse QID group from a medical database having flu, HIV and tick bites as diagnoses. This is much more desirable as having three different types of cancer in a QID group. Treating semantically close values as they were the same during data processing can be a necessary step to make l-diversity less delicate [4].

4.1.3 t-closeness

t-closeness [14] is a recently formalized constraint which mathematically limits the disclosure of less expected probability distributions. *Less expected* means showing clear deviation from the global distribution which an attacker might expect after looking at the overall data. As an example, consider a medical database containing diagnoses as sensitive attribute. If the overall distribution of HIV is small ($\leq 10\%$) then identifying a QID group containing 80% HIV diagnoses drastically increases the chance of any individual in question of having HIV. Although they can plausibly deny that fact (given l-diversity for $l \geq 2$), credibility drastically decreases in this case, especially in the eye of insurance companies, employers, etc.

Given a *KL-Divergence* based t-closeness constraint means that for each QID group G and each possible sensitive value $s_i \in S$ the inequality

$$t(G) = \sum_i P(s_i) ln \left(\frac{P(s_i)}{P(s_i|G)} \right) \leq t \tag{1}$$

must hold, where $P(s_i)$ is the global probability of seeing s_i and $P(s_i|G)$ the probability of s_i occurring in the specific QID group G. The undefined term $0 * ln(0)$ is set to 0.

It has been shown in [14] that the condition

$$\sum_i P(s_i|G) log(\frac{1}{P(s_i)}) \leq t + l \tag{2}$$

ensures both *t-closeness* and *l-diversity* at the same time. This combination is an actual state-of-the-art privacy constraint for data publishing.

4.2 Mechanics of differential privacy

The core assumption of differential privacy is that any individual's privacy is protected if its contribution does not change the outcome of an aggregated result significantly. This assumption has several important implications:

- The same aggregated result could be generated by either the original database or by a database where a single individual contribution is missing. This guarantees plausible deniability for each single person (but not for groups) no matter what background knowledge a potential attacker might have.
- If the outcome did not change **at all** between a database and the same database with one individual removed, then backward induction shows that the result would have always been the result from the empty database and therefore of no utility. So there is a clear tradeoff between individual privacy and utility of the outcome.

- If an individual's contribution does not change the outcome significantly, then the aggregated result can be protected against a single malicious or untruthful contribution.

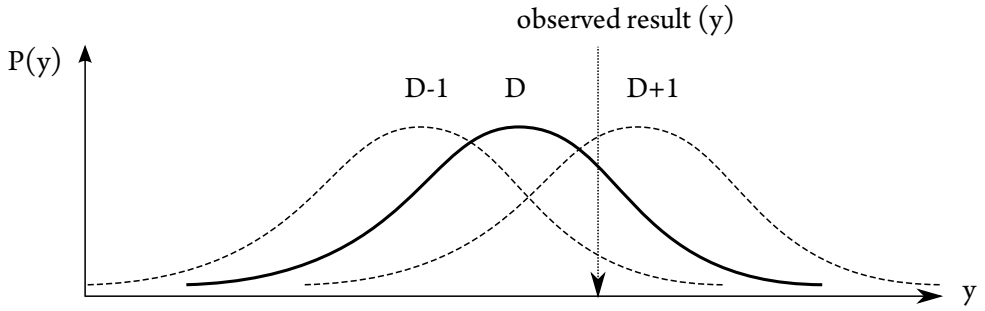

Figure 5: The observed output of the randomized $f(D)$ (PDF depicted here) is practically indistinguishable from the output the neighboring databases could generate. With small probability the observed result could even originate from a very distant database

4.3 ϵ-differential privacy

In order to formalize the above statements, the change in the outcome introduced by a single individual has to be quantified. *ϵ-differential privacy* [9] takes a probabilistic approach modeling an aggregate result $M(D)$ over database D as PDF $Pr[M(D) = y]$, as an aggregation mechanism M is expected to randomly generate multiple values of y with differing probabilities (noisy function). The outcomes over two databases D and D_Δ differing in a single entry can be considered similar enough if:

$$\frac{Pr[M(D) = y]}{Pr[M(D_\Delta) = y]} \leq e^\epsilon \tag{3}$$

Given ϵ, a mechanism M satisfying this constraint for each D is called ϵ-differentially private.

From the perspective of an attacker this constraint guarantees that, independently of what prior probability distribution (background knowledge) an attacker has, its posterior distribution after seeing a result is not significantly influenced by whether an individual contributed to the result or not. This is closely related to what t-closeness tries to guarantee at the original data level.

The exponential notation is sometimes replaced by a factor $\alpha = e^\epsilon$ and called α-differential privacy [11].

4.3.1 Sensitivity

How much M must be randomized to satisfy ϵ-differential privacy mainly depends on the change a single individual can cause to a result of the underlying truthful function f. As an example, consider a function f counting individuals which satisfy some predicate: An individual participating in the database can cause a maximum change of 1 to the outcome of f, so randomization must plausibly explain some change around

$f(D) \pm 1$. But if we consider a *sum* function over sensitive values to which an individual may contribute up to $1,000,000$, then changing the outcome by ± 1 clearly does not hide the contribution of such a large value, while a change of $\pm 1,000,000$ definitely does.

The maximum impact an individual can have on a function f is called *sensitivity* [9] $S(f)$:

$$S(f) = max(|f(D) - f(D_\Delta)|) \tag{4}$$

Sensitivity is additive in the sense that if $f(D) = (g(D), h(D))$ then $S(f) = S(g) + S(h)$, so multiple aggregations sum up their sensitivity[10].

The sensitivities of some common aggregation functions are:

- **1** for the *count* function or *histograms*.
- $max(S)$ for sums over arbitrary subsets of S. Notice that this is neither the maximum of the actual data, nor the maximum of the subset, but the maximally contributable number at all.
- $max(S)+1$ for average as it can be composed from the result of a sum and a count.

The implicit assumption in this sensitivity model is that individuals are independent, which may lead to problems when applied to real-world input [12].

4.3.2 The Laplace mechanism

Given a truthful function f with its sensitivity $S(f)$ an ϵ-differentially private mechanism M can be derived by adding noise sampled from a *Laplace* distribution with standard deviation $S(f)/\epsilon$: [9]

$$M(D) = f(D) + x \tag{5}$$

$$x \sim Lap(S(f)/\epsilon) \tag{6}$$

The Laplace distribution $Lap(\lambda)$ over a random variable X means that the probability $Pr[X = x]$ that X takes a value x is modeled as:

$$Lap(\lambda) : Pr[X = x] = \frac{1}{2\lambda} e^{\frac{-|x|}{\lambda}} \tag{7}$$

The diversity parameter λ is proportional to the standard deviation of this distribution. Both a lower ϵ (stricter privacy criterion) and a higher sensitivity broaden the standard deviation.

4.3.3 Limitations

Achieving ϵ-differentially private mechanisms using Laplacian noise is a relatively easy approach, but also has several limitations and privacy impacts:

- **suboptimal utility:** It has not been shown that the approach maximizes utility for any utility model. (Depending on the utility model, there are indeed better approaches, such as geometric noise for count queries) [12].
- **unbounded results:** Results may exceed the range of plausible values with small probability, truncation may be needed [12].

- **non-numeric results:** The Laplace mechanism does not make any sense on non-numeric or non-continuous outputs (consider randomizing a ZIP code or string) [15].
- **additive noise across multiple queries:** When the database is about to interactively answer multiple queries, the same additivity applies as for the sensitivities of simultaneous, non-interactively released results. The noise added to each single answer must scale with the sensitivity of all future answers (e. g. answering two counting queries requires each of the results to have twice as much noise as a single result). This effectively limits the number of answers a database with fixed ϵ may give or it additively increases ϵ [10, 16, 9]. If noise is not scaled properly, an attacker issuing arbitrarily many questions can eventually average out the noise.
- **How to set ϵ:** k-Anonymity has been around for a while now. There are established rules for censuses on what to generalize in which ways to almost certainly achieve $k > 1000$ without loosing too much information that could be useful for political and economical decision-making. However, there is no useful real-world experience on the choice of ϵ or α [12].
- **independent individuals:** The sensitivity model assumes that individuals are independent, which they roughly are in many controlled studies but certainly not in social networks [12]. In real life, an influential individual participating in a study could induce peer pressure, so identifying the aggregated contributions of his peer group (again, only individuals are protected, not groups) almost certainly reveals that he participated.

4.4 Utility model for differential privacy

4.4.1 Statistical utility measures

(Dis)utility can be measured trivially in terms of *variance* or *standard deviation* on the answer when the true answer is assumed to be the expected baseline. If the result is a histogram, the expected *earth mover's distance* can be used, if a probability distribution is constructed, the expected *KL-divergence* or *mutual information* can measure (dis)utility.

However, none of these statistical methods takes into account a potentially corrective effect of the user's background knowledge or any user-specific loss function, e. g. a binary loss (the data becoming fully useless or the consequences fatal when the answer exceeds some threshold). Recently, a general utility model for (discrete) differentially private mechanisms has been developed, which considers a user's loss function and background knowledge [11].

4.4.2 Universal utility measure

Suppose that each user might have an individual loss function $l(i, j)$ over the natural numbers. The function quantifies the loss a user experiences if he has seen a result j but the true result was i. This loss-function should only depend on i and $|i - j|$ and is non-decreasing with increasing $|i - j|$ (e. g. $|i - j|$ or $(i - j)^2$).

The user in question also has some background knowledge, which is modeled as a probability p_i over all possible answers i. e. g. if the user was about to interact with a database containing 5 records and he knew that either all or none of 5 records would

match his *count* query, the probability for 0 and 5 would be 1/2 and any other set to 0. In order to satisfy its belief, the user has some mechanism Y in mind, which remaps any output from the algorithm to the belief model of the user. In the last example, this function could map any result < 3 to 0 and any result ≥ 3 to 5, but any probabilistic mapping is possible, not only functions.

Given a randomized differentially private mechanism $M(i)$ where i is the real result, and the user's remap-mechanism Y, we can define a new mechanism $Z = M \circ Y$ by chaining both mechanisms. Given the probability $Pr[M(i) = y]$ that M delivers output y if the truthful answer was i and the probability $Pr[Y(y) = j]$ that the user interprets the output y as j we can compute the probability that the user perceives j if the real answer was i by summing up the decision paths over all possible y:

$$z_{ij} = Pr[Z(i) = j] = \sum_y Pr[M(i) = y] \cdot Pr[Y(y) = j] \tag{8}$$

Having these probabilities, we can compute the expected loss $L(i)$ with i being the true result after applying both mechanisms:

$$L(i) = \sum_j z_{ij} \cdot l(i, j) \tag{9}$$

Aggregating expected losses is done by weighting it with the user's belief that each answer i has probability p_i and summing up the expected losses.

$$\begin{aligned} L &= \sum_i p_i \cdot L(i) && (10) \\ &= \sum_i p_i \cdot \sum_j z_{ij} \cdot l(i, j) && (11) \end{aligned}$$

We now know that we can maximize utility by minimizing L during the design phase of our privacy mechanisms. The remarkable property of this model is, that it is the first to quantify the utility of actual mechanisms (e. g. the Laplacian mechanism) while not assuming anything about the underlying data (this property is referred to as *obliviousness*).

4.4.3 Consequences of the universal utility model

Literature [11] has shown that differential privacy can indeed use better performing distributions than Laplace noise with respect to this utility model, e. g. by applying noise sampled from a geometric distribution to queries with sensitivity 1 (count queries). In the discrete case, the difference Δ which is added to the output is sampled from the following geometric probability distribution:

$$Pr[\Delta = x] = \frac{1 - \alpha}{1 + \alpha} \cdot \alpha^x \tag{12}$$

The parameter α is exactly the privacy level in the definition of α-differential privacy or equal to e^ϵ in ϵ-differential privacy. With respect to the universal utility model this noise is optimal over all legal loss functions a user might have.

5 Future Work

Concerning generalization-based approaches, there are several inherent problems that need to be addressed:

- The utility vs. privacy problem is of greater concern in the context of generalization-based approaches as in differential privacy, because they do not depend on some fixed aggregation or mechanism. Especially in the domain of pattern recognition and machine learning utility studies are missing [14].

- Updating the published data is very dangerous as the difference may reveal contributions of added or removed individuals. Although m-invariance is a new constraint which allows republication given a limited background knowledge (that the publisher has to know in advance) it does not scale to arbitrary background knowledge. Alternatives need to be explored [1].

- High dimensionality decreases the utility of the data tremendously. This is due to the fact that quasi-identifiers easily become unique again if an attribute/dimension is added and a lot of values need to be suppressed in the end. This *course of dimensionality* [17] has to be broken to make these approaches viable for tracking data, social networks, genetic information and similar data with thousands of dimensions.

Differential privacy may suffer from similar problems concerning updates, but the impact on updating the database has not been studied yet. Some open questions can be deduced straightforwardly from the limitations outlined in 4.3.3, most notably:

- The individual independence assumption has to be evaluated for real-world scenarios, e. g. how well the sensitivity-based model still works in social networks [12].

- The (only) universal utility model for count queries needs to be generalized to fit more query types. This may further improve the privacy-utility-tradeoff as the model can be seen as objective function one can try to minimize by choosing alternative randomization methods [11].

- The first approach of creating an ϵ-differentially private k-anonymization uses random sampling. The researchers expect that there might be other or better algorithms to achieve ϵ-differential privacy guarantees for k-anonymity [13].

- Differential privacy has been generalized to arbitrary domains for a limited type of mechanisms (e. g. auctions) [3]. It is subject to further research what other mechanisms and algorithms can be made differentially private, e. g. mechanisms that generate trees, compound objects or complex classifiers.

There are also some exotic approaches that use compressive sensing [18], which is basically a recently discovered lossy compression capturing the principal components of the data but omitting very individual information and therefore satisfying strong privacy guarantees, even ϵ-differential privacy. Further research may signalize whether these approaches are a feasible alternative to adding noise.

6 Conclusion

Differential privacy clearly targets privacy-preserving data mining (PPDM) as it focuses on the actual aggregation, while indistinguishability-based approaches (k-anonymity, etc.) operate on the syntactic level of the underlying data, therefore being more suitable for privacy-preserving data publishing (PPDP). Differential privacy emerged from the need for a more rigorously provable privacy guarantee than any historic approach including those in the area of PPDP, so the lines of research were split around 2007. It turned out that modern PPDP approaches, such as t-closeness, are indeed challenging to develop further and getting a better tradeoff between privacy and utility, while differential privacy is constantly making (at least mathematical) progress.

Nowadays, the choice between these two lines heavily depends on the expected use case. Published real-world data can have an extreme impact on the productivity in this area, as it motivates the development of new mechanisms that deliver deep insights or business value (e. g. recommender systems after Netflix released movie-ratings issued by a large set of users). Aggregated results, especially when released in a differentially private manner, have a stronger guarantee that nobody's privacy will ever be harmed but are effectively useless when the user of the data wants to build new analysis methods. They have a more or less informative character or can be seen as a tool which can be built into existing algorithms to protect their users. Nevertheless both lines have a place alongside each other and new research connecting both areas again seems to be coming up.

In any case, there is still a lot of work ahead before the mathematical models fit some current and future real-world scenarios, particularly those concerned with tracking data, communication (meta-)data, social networks, genetic information and biometric data.

References

[1] X. Xiao and Y. Tao, "M-invariance: towards privacy preserving re-publication of dynamic datasets," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 689–700.

[2] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, Jan. 2008.

[3] F. Mcsherry, "Mechanism design via differential privacy," in *Proceedings of the 48th Annual Symposium on Foundations of Computer Science*, 2007.

[4] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Comput. Surv.*, vol. 42, no. 4, pp. 14:1–14:53, Jun. 2010.

[5] N. R. Adam and J. C. Worthmann, "Security-control methods for statistical databases: a comparative study," *ACM Comput. Surv.*, vol. 21, no. 4, pp. 515–556, Dec. 1989.

[6] M. H. Hansen, "Insuring confidentiality of individual records in data storage and retrieval for statistical purposes," in *Proceedings of the November 16-18, 1971, fall joint computer conference*, ser. AFIPS '71 (Fall). New York, NY, USA: ACM, 1971, pp. 579–585.

[7] D. E. Denning, "Secure statistical databases with random sample queries," *ACM Trans. Database Syst.*, vol. 5, no. 3, pp. 291–315, Sep. 1980.

[8] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, Oct. 2002.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Third conference on Theory of Cryptography*, ser. TCC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 265–284.

[10] C. Task, "A practical beginners' guide to differential privacy," http://www.cerias.purdue.edu/news_and_events/events/security_seminar/details/index/j9cvs3as2h1qds1jrdqfdc3hu8, 2012, accessed: 2013-12-12.

[11] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," in *Proceedings of the 41st annual ACM symposium on Theory of computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 351–360.

[12] C. Clifton and T. Tassa, "On syntactic anonymity and differential privacy," in *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*. Brisbane, QLD: IEEE, 2013, pp. 88–93.

[13] N. Li, W. H. Qardaji, and D. Su, "Provably private data anonymization: Or, k-anonymity meets differential privacy," *CoRR*, vol. abs/1101.2604, 2011.

[14] C. Sha, Y. Li, and A. Zhou, "On t-closeness with kl-divergence and semantic privacy," in *Proceedings of the 15th international conference on Database Systems for Advanced Applications - Volume Part II*, ser. DASFAA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 153–167.

[15] C. Dwork, "Differential privacy: a survey of results," in *Proceedings of the 5th international conference on Theory and applications of models of computation*, ser. TAMC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–19.

[16] W. Qardaji, "Differentially private publishing of geospatial data," http://www.cerias.purdue.edu/news_and_events/events/security_seminar/details/index/9nr7je9aqbqneqem9hrg2salic, 2013, accessed: 2013-12-12.

[17] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st international conference on Very large data bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 901–909.

[18] Y. D. Li, Z. Zhang, M. Winslett, and Y. Yang, "Compressive mechanism: Utilizing sparse representation in differential privacy," *CoRR*, vol. abs/1107.3350, 2011.

Private Information Retrieval

Cloud Security Mechanisms Seminar – Summer Term 2013

Maximilian Schneider

1 Introduction

A Private Information Retrieval Protocol allows an user to retrieve an item from a server in possession of a database without revealing any information about which item is retrieved. In simple terms: Bob is in possession of a database $x = \{0; 1\}^n$ from which Alice wants to read the i-th bit x_i without Bob knowing the index i in which Alice is interested in.

Possible Applications for this might be that Bob provides access to a database of Chemical structures. Alice is interested in the properties in specific parts of a new structure she synthesized. Using a PIR protocol she would be able to look up these specific parts in Bob's database without revealing to him what she is currently researching in.

A trivial solution to this problem would be for Alice to download the entire database. Assuming that the server would only respond with a fraction of the database, one can argue that in this case the server would know which items were not not retrieved and therefore gain information about which item was retrieved. This implies that the trivial approach is also the optimal solution to Private Information Retrieval from a single server.

In the research field of Information Theoretic Private Information Retrieval the existence of k non-co-operating replications of the database is assumed. This assumption is sometimes weakened to the existence of k replications of the database of which up to t may co-operate, which is called t-private Information Retrieval.

Another notable branch of research is Computationally Private Information Retrieval. To the problem definition of Private Information Retrieval the assumption is added, that the server is polynomially bounded. Contrary to Information Theoretic Private Information Retrieval this allows protocols to involve only a single server.

Computationally Private Information Retrieval problems can be additionally constrained, so that the user is allowed to receive only information about the particular index he queried. This variation is called Symmetric Private Information Retrieval [1].

In the following the primary branches of research, namely Information Theoretic Private Information Retrieval and Computationally Private Information Retrieval, are presented. The respective history of each field of research and the connection to research of other cryptographic primitives will be discussed. Further on the findings of [2] and [3] are explained in detail as they were of considerable impact to this field of research.

2 Foundations

2.1 Oblivious Transfer and SPIR

Definition 1 *In a 1-out-of-n oblivious transfer protocol the party called "the sender" has n messages, but wishes to share only one of those. The other party, the receiver, wants to receive the message corresponding to an index i, which should remain hidden from the sender.*

One of the requirements for a Private Information Retrieval protocol is that the communication must be sublinear. Therefore every SPIR protocol is always also a 1-out-of-n oblivious transfer protocol. Moreover one can arbitrarily transform single database CPIR protocols into SPIR protocols [4]. This implies that every single database CPIR protocol is also a 1-out-of-n oblivious transfer protocol.

2.2 Locally Decodable Codes

Definition 2 *A q-query locally decodable code encodes an n-bit message x by an N-bit codeword C(x) such that any bit x_i of the message can be probabilistically recovered by querying only q bits of the codeword, even if some constant fraction of the codeword has been corrupted [5].*

The research of Private Information Retrieval has caused advances in the field of Locally Decodable Codes, and vice versa. This was to some extent only possible because because ITPIR resembles receiving Locally Decodable Codewords.

Theorem 1 *If the queries to k different servers correspond to the q extracted bits from a Locally Decodable Codeword, and if privately retrieving x_i corresponds to recovering x_i, then from every PIR scheme a LDC scheme with constant probability 1 can be deduced.*

3 History of Information Theoretic Private Information Retrieval

The Problem of Private Information Retrieval was first proposed by [2] in 1995. To improve upon the communication complexity (measured in bits exchanged between client and server) they tried to replicate the database on completely non-co-operating servers. Their scheme builds upon the idea, that they construct uniformly, randomly selected subsets of the index set as queries, which do not reveal any information about the retrieved index. For the specific case of 2 databases their result, which exchanges a total of $O(n^{1/3})$ bits between the client and all servers in a single round-trip, is considered as optimal.

[6] improved upon their solution by changing the emulation. The one proposed by [2] relied on covering codes of a distance 1. If a non-perfect covering code is used, certain codewords are covered twice which leads to redundant computation and communication. The improvement relied upon a recursive definition to distribute the emulation and resulted in scheme with communication complexity $O(n^{\frac{1}{2k-1}})$.

Roughly five years later this result was first improved upon by [7]. Their solution was based on representing the database as a multivariate polynomial over a finite field. By sending each server only subsets of the substitutions for the polynomial's free variables, it can reduce the polynomial only to contain at least one free variable. Their result was not only a major contribution to the field of Private Information Retrieval, but rather also advanced the research of Locally Decodable Codes.

[8] presented an improvement in Locally Decodable Codes using Mersenne Primes. Though Locally Decodable Codes have a probabilistic nature, for this particular code a construction for a non-probabilistic three server Private Information Retrieval protocol was given. Assuming Infinite Mersenne Primes the total communication complexity is $O(n^{\frac{1}{loglogn}})$. The resulting communication complexity of $O(n^{\frac{1}{32582658}})$ results from using the largest known Mersenne Prime.

4 History of Computationally Private Information Retrieval

The Problem of Computationally Private Information Retrieval was first introduced by [9]. They presented a modification to the original Information Theoretic Private Information Retrieval solution by [2] which still relied on 2 non communicating servers. Depending on the security parameter $e > 0$, the scheme's communication complexity was $O(n^e)$.

In the same year [3] proved that CPIR schemes do not need to rely on non-co-operating replications. They presented the first CPIR scheme which only involved communication with a single database. Their solution was based on the Goldwasser-Micali Cryptosystem [10] which supports homomorphic exclusive or.

Subsequently many different CPIR protocols based on various other homomorphic crypto-systems have been discovered. Most notably the result by [11] which was based on the Damgård-Jurik crypto-system with a sublinear communication complexity of $O(loglogn)$.

Most recently schemes based on the hardness of the hidden lattice problem [12] have been published. These have been the first schemes which actually fast enough to be put into actual application.

5 Mechanism Details

5.1 2-Server Information-Theoretic Private Information Retrieval

The scheme proposed by [2] relies on constructing multiple different subsets of the index set as queries. In order to hide the retrieved index i, each of these subsets is selected uniformly at random. The replications reply with the exclusive or of all selected bits. Through their construction of the subsets they can ensure that the retrieved index i is only included in an uneven number of subsets while every other index is included in an even number of subsets. Therefore the associativity of the exclusive or operation allows to compute the retrieved index from the single bit responses of all replications.

As an example for the case of two replications the respective queries Q and $Q*$ would be constructed as subsets of the index set $I = [1; n]$:

$$Q \subset I, \text{where } \forall j \in I : P(j \in Q) = 0.5 \tag{1}$$

$$Q* = \begin{cases} Q \setminus i, & \text{if } i \in Q \\ Q \cup i, & \text{otherwise} \end{cases} \tag{2}$$

$$\mathcal{R}(Q) = \bigoplus_{j \in Q} x_j \tag{3}$$

As specifying a single subset of the index set in a naïve way requires n bits (one for each bit in the queried database), they instead chose to represent their index set as a d-dimensional cube. A simple example representation of x for $d = 2$ and $n = 9$ would be:

$$x = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix} \tag{4}$$

They adapted their query construction to specify d subsets of $[1;l]$ where $l = \lceil \sqrt[d]{n} \rceil$ which reduced their query size from n to $d \cdot \sqrt[d]{n}$. This modification increased the number k of required replications to $k = 2^d$. The four queries for the last example from equation 4 would be constructed with the help of S_1 and S_2, each randomly selected subsets of $[1;l]$.

$$S_d* = \begin{cases} S_d \setminus i_d, & \text{if the } d\text{th component of } i, i_d \in S_d \\ S_d \cup (i_d), & \text{otherwise} \end{cases} \tag{5}$$

$$Q_1 = Q(S_1, S_2) = \{(s_1, s_2) \mid s_1 \in S_1 \wedge s_2 \in S_2\} \tag{6}$$

$$\text{accordingly: } Q_2 = Q(S_1, S_2*), Q_3 = Q(S_1*, S_2) \text{ and } Q_4 = Q(S_1*, S_2*)$$

As the increased number of replications also implied increased amount of communication, they let some replications emulate others, thereby reducing the number of needed replications and total communication again. In order to guarantee privacy the respective queries for the emulated replications may not be provided to the emulating replications. While this procedure increases the amount of bits sent back from each server, it also reduces the number of servers and the total communication from the client to all servers, therefore it is called balancing.

The emulating replications prepare responses for every possible query, which one of the emulated replications could have received, by modifying the query transmitted to them. Their construction based on perfect binary covering codes results in an optimal solution when using 2 codewords of length 3 to construct an emulation scheme for 3-dimensional addressing in which 2 servers each emulate 1 additional replication per dimension. In that case one server would receive a query $Q(S_1, S_2, S_3)$ while the other one receives $Q(S_1*, S_2*, S_3*)$. The first one would have to guess S_1*, S_2* and S_3* in order to calculate the missing queries, it should emulate:

$$Q(S_1*, S_2, S_3), Q(S_1, S_2*, S_3) \text{ and } Q(S_1, S_2, S_3*)$$

As no information regarding those sets can be revealed, as this would let the first server gain information on i, every possible S_d* has to be treated equally and a responses for each of them is calculated. The same is also true for the second server.

This leads to a 2 Server Protocol, in which the client sends each server a query of $3 \cdot \sqrt[3]{n}$ bits and receives from each a response of $1 + 3 \cdot \sqrt[3]{n}$ bits this results in a total communication of $12\sqrt[3]{n}$ bits per retrieved bit.

5.2 Single Server Computationally Private Information Retrieval based on Goldwasser-Micali

The scheme proposed by [3] mainly relies on the Goldwasser-Micali cryptosystem [10] which support homomorphic computation of the exclusive or. The basic idea is that the server only executes homomorphic computations on encrypted numbers and that only the client can decrypt the input and result of the computation so that his privacy is ensured.

In order to retrieve bit i, the client sends initially a series of uniformly at random chosen cyphertexts $E_1, ..., E_n$ having the following property:

$$E_j = \begin{cases} \mathcal{E}(1), & \text{if } j = i \\ \mathcal{E}(0), & \text{otherwise} \end{cases} \tag{7}$$

Computing uniformly selected cyphertexts to encrypt 0 is easily possible, as the Goldwasser-Micali cryptosystem relies on the quadratic residuosity problem, first discovered by [13]. The Server in turn replies with the response $\mathcal{R}$, which is computed as follows:

$$\mathcal{R}(x, E_1, ..., E_n) = \prod_{j=0}^{n} \begin{cases} E_j^2, & \text{if } x_j = 1 \\ E_j, & \text{otherwise} \end{cases} \tag{8}$$

As multiplication of cyphertexts corresponds to exclusive or of plaintexts in the Goldwasser-Micali cryptosystem the following holds true: $\mathcal{E}(0) \cdot \mathcal{E}(0) = \mathcal{E}(0 \oplus 0) = \mathcal{E}(0)$. Therefore squaring E_j or not for all $j \neq i$ has no influence on the decrypted plaintext of the product. Furthermore the result of the decryption $\mathcal{D}(\mathcal{R})$ of the computation can be simply be represented as:

$$\mathcal{D}(\mathcal{R}) = \mathcal{D}\left(\prod_{j \neq i} \mathcal{E}(0) \cdot \begin{cases} \mathcal{E}(1) \cdot \mathcal{E}(1), & \text{if } x_i = 1 \\ \mathcal{E}(1), & \text{otherwise} \end{cases}\right) = 1 \oplus x_i \tag{9}$$

Similiar to the protocol described in 5.1 the total communication between the client and server is linear to the database size, when implementing the protocol in a naïve way. Therefore [3] proposed a recursive extension to their protocol. The basic idea is that the database is partitioned into chunks of the size $n_L = \left\lceil \sqrt[L]{n} \right\rceil$ for a given recursion level of L. The basic protocol is then applied to each of these n_L^{L-1} chunks C_c using the same cyphertexts $E_1, ..., E_{n_L}$ creating the base for the recursion.

$$C_c = \{x_j \mid j \in \big((c-1) \cdot n_L; c \cdot n_L\big]\}, \text{ where } c \in [1; n_L^{L-1}] \tag{10}$$

$$R_{1,c} = \mathcal{R}(C_c, E_1, ..., E_{n_L}) \tag{11}$$

The n_L^{L-1} results serve as a base for the recursion in the case of recursion level $l = 1$. To execute the next step of the recursion the results from the previous step are splitted into n_L chunks of size n_L^{L-l}, where l is the next recursion level. Over each of those chunks the PIR protocol is evaluated again, leading to the following computation:

$$\forall l \in [2; L] : R_{l,c} = \prod_{r=1}^{n_L} \begin{cases} E_{(l-1)\cdot n_L + r}^2, & \text{if } R_{l-1, c\cdot n_L + r} = 1 \\ E_{(l-1)\cdot n_L + r}, & \text{otherwise} \end{cases} \tag{12}$$

With each invocation of the recursion the size of the result set decreases until $l = L-1$ and only n_L results are left. Applying the recursion another time at this point, would reduce the size of the result set to a single number, but it would require n^L additional cyphertexts to be transmitted from the client to the server. As the total communication would therefore increase this step is omitted and the n_L results are transmitted directly.

6 Recent Developments & Future Research

Private Information Retrieval is still an active field of theoretical research as even current schemes struggle to be incorporated into applications. This is mostly due to the low time efficiency when comparing the computation necessary for PIR schemes to transferring the whole database. In 2006 PIR was still considered practically infeasible at all [14] and the research community was criticized for focusing only on communication complexity and not regarding overall time efficiency.

This discussion has shifted the problem from achieving Private Information Retrieval with least communication possible to the development of algorithms which tradie increased communication complexity for reduced computation complexity [12, 15]. Since then PIR algorithm researchers started to consider the applicability of these algorithms to actual computation, including computation on GPU hardware [16].

These new algorithms have increased the confidence in the feasibility of PIR [17] and led to the first applications based on Private Information Retrieval. One notable example is a Privacy-Preserving Domain Name System [18] which combines distributed hash tables and a CPIR scheme [19] to provide private lookup of DNS names.

Since the emergence of common and feasible anonymization technologies like the TOR network [20] ITPIR has become an even more interesting topic of research. Due to the constraint of non-cooperating replications ITPIR was always regarded impractical for applications. But when considering that the identity of the requesting party can be hidden from the different replications, this also means that cooperating becomes impossible for these replications.

7 Conclusion

Since the first publication dealing with Private Information Retrieval nearly 20 years have passed. Only current advances in research have been incorporated into practical applications, primarily due to the high overhead in using PIR protocols. Another reason might be that the consequences of not using PIR in most use cases usually are weighted irrelevant, as the provider of information is often trusted to deal with the data of it's users in a sensible way.

The strong ties between Information Theoretic Private Information Retrieval and Locally Decodable Codes allow PIR in general to benefit from development in this related area, the same holds true for Computationally Private Information Retrieval in the case of Oblivious Transfer. The continuing research and following improvements, as well as the increasing adoption into applications promise further development in the future.

References

[1] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting data privacy in private information retrieval schemes," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ser. STOC '98. New York, NY, USA: ACM, 1998, pp. 151–160.

[2] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, pp. 965–981, Nov. 1998.

[3] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval," in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, ser. FOCS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 364–373.

[4] G. Di Crescenzo, T. Malkin, and R. Ostrovsky, "Single database private information retrieval implies oblivious transfer," in *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, ser. EUROCRYPT'00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 122–138.

[5] Wikipedia, "Locally decodable code — wikipedia, the free encyclopedia," 2013, [Online; accessed 22-September-2013]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Locally_decodable_code&oldid=571624006

[6] A. Ambainis, "Upper bound on the communication complexity of private information retrieval," *Automata, Languages and Programming*, pp. 1–9, 1997.

[7] A. Beimel and Y. Ishai, "Breaking the barrier for Information-Theoretic Private Information Retrieval," *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 1–30, 2002.

[8] S. Yekhanin, "New locally decodable codes and private information retrieval schemes," *Electronic Colloquium on Computational Complexity*, vol. 127, 2006.

[9] B. Chor and N. Gilboa, "Computationally Private Information Retrieval," *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 304–313, 1997.

[10] S. S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, ser. STOC '82. New York, NY, USA: ACM, 1982, pp. 365–377.

[11] H. Lipmaa, "An oblivious transfer protocol with log-squared communication," *Information Security*, 2005.

[12] C. Aguilar-Melchor and P. Gaborit, "A lattice-based computationally-efficient private information retrieval protocol," *Cryptol. ePrint Arch., Report 446*, 2007.

[13] C. F. Gauß, *Disquisitiones Arithmeticae*, 1801.

[14] R. Sion and B. Carbunar, "On the Computational Practicality of Private Information Retrieval," *Proceedings of the Network and Distributed Systems Security Symposium 2006*, 2006.

[15] J. Trostle and A. Parrish, "Efficient computationally private information retrieval from anonymity or trapdoor groups," *Information Security*, 2011.

[16] C. A. Melchor, B. Crespin, P. Gaborit, V. Jolivet, and P. Rousseau, "High-speed private information retrieval computation on gpu," *SECURWARE'08. Second International Conference on Emerging Security Information, Systems and Technologies, 2008*, pp. 263–272, 2008.

[17] F. Olumofin and I. Goldberg, "Revisiting the computational practicality of private information retrieval," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 158–172.

[18] Y. Lu and G. Tsudik, "Towards Plugging Privacy Leaks in Domain Name System," *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–10, 2010.

[19] C. A. Melchor and P. Gaborit, "A fast private information retrieval protocol," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1848–1852.

[20] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, p. 21.

Trust-Based Access Control

Cloud Security Mechanisms Seminar – Summer Term 2013

Vincent Schwarzer

1 Introduction

Through the increase of interconnected information systems often called ubiquitous computing and the offering of services by them to other peers in the network new challenges have arisen. These are rights management and how to handle unknown peers as well as how to find trustworthy service providers in a flood of similar offerings. An approach to tackle these challenges is to introduce *Trust and Reputation Systems* (TRS) where *Trust - Based Access Control* (TBAC) is a part of. TRS can be used as decision support systems for *Software-as-a-Service* (SaaS) offerings or support rights management by granting and revoking access rights based on trust/reputation scores.
In a SaaS environment TRS help service consumers to choose a service provider by providing a scoring for each provider based on ratings from peers who had transactions with the service provider before. Today service consumers often does not have sufficient information about the quality of a service provider offers what trust and reputation systems could compensate.
Similar problems exist in ubiquitous computing that build large dynamic networked infrastructures where new unknown peers (Information systems/users) want to access resources. TRS help the peer to find other trustworthy peers or to assign individual privileges to each peer autonomous because doing this manually is no longer sufficient. Using TBAC minimizes the efforts necessary for rights management because the whole rights management is automated and needs only minimal intervention by the provider.

TBAC introduces the trust and reputation in the field of access control mechanism. Based on feedback after each transaction between peers a trust/reputation scores is computed which grants/revokes access rights.These scores are determined for each peer individually and are based on different factors like behavior or context (place, time).

In this paper I will give a general overview over TRS and introduce TBAC especially.

2 Foundations

To get an understanding for Trust and Reputation System and TBAC it is important to introduce some central terms used in this research field.

2.1 Trust

Trust has been researched by many researchers over the years [1, 2] but there is no unified definition of trust. The reason for that is that the definition of trust is subjective and heavily depends on the authors viewpoint. Two commonly used definitions for trust are:

Reliability Trust by Gambetta Trust is the subjective probability by which an individual A, expects that another individual B, performs a given action on which its welfare depends [3].

Decision Trust by McKnight & Chervany Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible [4].

The amount of trust necessary to conduct a certain action depends on the *risk* involved. A trust score can be seen as a measure how trustworthy a peer acts in future transactions.

2.2 Risk

Risk is defined as the combination of likelihood and severity of an accident according to Dimmock [5]. To determine the risk of an action TRS use probabilistic cost-benefit analysis to determine if an action or privilege should be granted or denied.

2.3 Reputation

Reputation is defined according to the Oxford Dictionary [6] as: "Reputation is what is generally said or believed about a person's or thing's character or standing.". Generally, reputation can be seen as a collective measure of trustworthiness based on rating or referrals from other peers. A Reputation score can relate to a group or to a single peer.

2.4 Distinction Trust & Reputation

Jøsang et. al. [7] published an article "A Survey of Trust and Reputation Systems for Online Service Provision" where they examined TBAC and the difference between Trust and Reputation. They developed an example that should illustrate the difference between the two terms.
(1) "I trust you because of your good reputation."
(2) "I trust you despite your bad reputation."

In the first sentence the peer is using public information to base his trust in the trustee. Compared to the second sentence where the relying peer has some private information about the trustee e. g. direct experience that overrule any reputation that a person might have. That means that trust weights more then reputation as notion.

2.5 Difference Trust System & Reputation System

As described by Jøsang et. al. [7], a distinction has to be made between *trust systems* and *reputation systems*. The main difference between the two systems is that a trust system score that reflects how the subjective view of the relying peer is about another peer. Reputation system score reflects the opinion about a peer seen by the whole community. Another difference is that trust transitivity is a explicit component in trust systems compared to reputation systems that only take it implicitly into account. As input trust systems take subjective and general measures of trust compared to reputation systems who usually use information or ratings about specific events like transactions.

2.6 Ubiquitous Computing

Ubiquitous Computing describes a computing concept first envisioned by Weiser [8] where computing is everywhere and anywhere. Entities can be information system of any kind like tablets, smartphones, cars or refrigerators. They are autonomous and often mobile and should be able to react to unforeseen circumstances like disconnection to interacting with other unknown entities. One of the main challenges in this environment is the management of access rights when unknown entities interact with each other.

2.7 Centralised & Distributed Reputation Systems

Reputation Systems can be distinguished into two types centralised and distributed reputation systems as described by Jøsang et. al. [7]. In a *Centralised Reputation System* a central authority collects all given ratings for all peers and computes a reputation score for each peer. To do this centralised communication protocols are used that participants can provide reputation ratings and obtain reputation scores from other peers. In a *Distributed Reputation System* is no central location for submitting and obtaining ratings. Therefore each peer stores their opinion about other peers and calculates the reputation score. If another peer wants to do a transaction with an unknown peer he can request reputation scores from other peers who already have transacted with the peer.

3 History

The term TRSTrust and Reputation systems where TBAC is a part of was first mentioned by Rasmussen & Jansson [9] in 1996. Where they used the term *hard security* for traditional access control mechanisms and authentication and *soft security* for social control mechanisms where TRS are a part of. Since then many different methods and technologies have been proposed how to compute trust like *Simple Summation or Average of Ratings*, *Fuzzy Systems* [10], *Bayesian Systems* [11, 12, 13], *Analytic Expressions* (EigenTrust [14], Peer-Trust [15]), *Discrete Trust Models* [16] as well as their application in different kind of information systems.
These information systems can be:

Role Based Access Control with notion of Trust [17, 18] This implementation is called TBAC which combines the traditional *Role-based Access Control* (RBAC) with TRS. Roles associated with the permissions are assigned dynamically based on the trustworthiness score of a peer. Based on this score the role of the peer changes continuously and thus his rights in the system. Example for this systems are *Trust and Context Based Access Control* (TCAC) [19], TrustBAC [20]

Peer-2-Peer Networks Through the anonymous and open nature of Peer-2-Peer networks TRS are used to identify trustworthy peers and exclude malicious peers that spread inauthentic files. Challenges in these systems are according to Kamvar et. al. [14] that TRS in P2P environment should be:

Self-Policing That the peers define and enforce good behaviour without a central authority in the system.

Anonymity The usage of TRS should not interfere with the anonymity of the peers in the network.

No Profit to newcomers There should be no benefit for being a new peer in the network to prevent malicious peers to change their identity. Reputation can be only gained through positive interactions with other peers.

Minimal Overhead There should be minimal overhead when a TRS is used.

Robust against malicious collectives The TRS should be robust against malicious behaviour of peers.

Multi Agent System Multi Agent Systems [21, 22] consist of intelligent and autonomous agents that interact together to solve problems efficient. Trust and reputation systems are used to find trustworthy exchange partners.

Researchers that are very active in the research of trust and reputation systems are Audun Jøsang and Nathan Dimmock.
The majority of the proposed models uses Trust Computation based on transaction ratings. Another less researched approach is calculating trust values through a certificate based system like the one proposed by Herzberg et. al. [23]. All models assume that some mechanism to identify the peers identity is in place. In the next section we will describe some trust and reputation systems more in detail.

4 Mechanism Details

Most of the current proposed *Trust and Reputation models* follow four general steps formulated by Marti & Garcia-Molina [24] in 2006 shown in 1. These are:

Collect Information In trust and reputation systems the transactional behaviour of each peer is collected to determine how trustworthy a peer is.

Aggregate Information The peer behaviour is scored and ranked based on the transaction history.

Select peer & Interact Based on the score and ranking a peer is chosen for the next transaction.

Punish & Reward Each peer rates the transaction with the other peer. Based on the rating the system can take action against malicious peers and reward trustworthy peers.

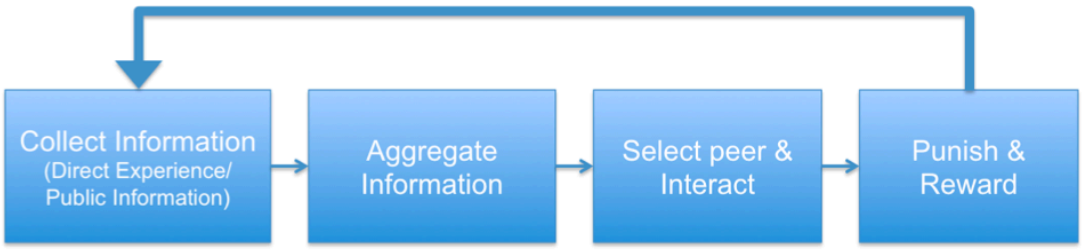

Figure 1: Steps Trust and Reputation Models cf. Marmol et. al. [25]

5 Examples for TBAC

5.1 eBay

A popular example for reputation systems is the online auction house eBay evaluated by Resnick et. al. [26]. In eBay's centralized reputation system buyer and seller can give ratings in different categories and a overall rating after each transaction. The reputation of each participant is calculated by aggregating the overall ratings for the last 6 months and ranks each member with different coloured stars. Additional to the rating each buyer/seller can leave a text comment giving more detail about his rating.

5.2 Stack Overflow

The website Stack Overflow is using a centralized proprietary TBAC system to manage the user privileges in their community called Stack Overflow Meta. The main idea behind the system is that each action in the community (create threads, answer to topics, etc.) the user receive for good behaviour or loses for malicious behaviour a certain amount of reputation points. These reputation points for each action are aggregated for each user to a reputation score with the formula seen in 1. Based on the score each user gets rights granted/revoked.

$$Reputation = \sum_{i}^{n} Rep(Action) \tag{1}$$

Every privilege in the community is bound to a certain amount of necessary reputation. The goals of the system are to act as incentive to be active in the community and to act as a filtering mechanism against malicious or misbehaving users.

5.3 TCAC

TCAC is a further development of TBAC proposed by Feng et. al. [19]. It extends TBAC system with context information thus privileges are based on trustworthiness and context information of the peer. The context characterize the situation of an entity which can be a person, time or location. These context constraints can be defined for each object.
An example for a TCAC system can be seen in the picture 2 where a peer u with the trust score t_u send an access request (AR) where the peer in session s wants to perform operation p on an object o. This request gets evaluated by the *Access Control System* (ACS). The ACS retrieves the peer information and checks if the peer has a sufficient trust score for the operation on the requested object. If this is the case the system checks if the peer satisfies the context requirements. If this is the case the peer gets dynamically a new role assigned and can conduct the requested operation on the object. If one of the checks fails the request will be denied. After the request, the system evaluates the transaction and stores the adjusted trust score for the peer in the database.

5.4 Beta Reputation System

The Beta Reputation System was proposed by Jøsang & Ismail [27] in 2002 as flexible framework for reputation services in e-commerce applications. The system is based on Bayesian Systems and the *Belief Model* proposed by Jøsang & Audun [28] in 2001 and combines the Reputation Function and Rating with Reputation Discount and Forgetting.

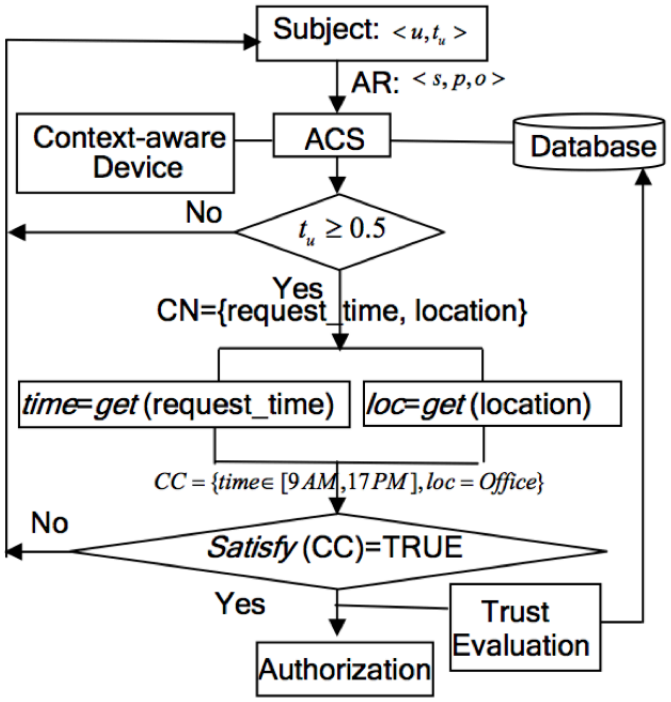

Figure 2: TCAC access request see Feng. et.al. [19]

5.5 Beta Density Function

The Beta Density Function is the mathematical foundation of the Beta Reputation System to represent the probability distribution of positive or negative behaviour. For the Beta Reputation System the beta distribution $f\left(p \mid \alpha, \beta\right)$ is used expressed by using the gamma function Γ:

$$f\left(p \mid \alpha, \beta\right) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{2}$$

With additional following restrictions $0 \leq p \leq 1$, $\alpha > 0$, $\beta > 0 and p \neq 0 if \alpha < 1 and p \neq 1 if \beta < 1$. Applying this formula to observing two possible outcomes e. g. positive and negative outcomes. Where r stands for positive feedback and s for negative feedback the beta density function of observing positive feedback r in the future can be expressed by setting:

$$\alpha = r + 1 \, and \, \beta = s + 1, \; where \, r, s \geq 0 \tag{3}$$

5.5.1 Beta Reputation Function and Rating

The Beta Reputation Function and Rating is the core of the Beta Reputation System. After each transaction both peers give feedback positive or negative. In the case of the Beta Reputation system the feedback is given as tuple *(r,s)* of continuous values where $0 \leq r \leq 1, 0 \leq s \leq 1, r + s = 1$. The initial values of r and s are r,s=0. Based on this the reputation function is defined as follows:

$$\varphi\left(p \mid r_T^X, s_T^X\right) = \frac{\Gamma\left(r_T^X + s_T^X + 2\right)}{\Gamma\left(r_T^X + 1\right)\Gamma\left(s_T^X + 1\right)} p^{r_T^X} \left(1-p\right)^{s_T^X}, where \; 0 \leq p \leq 1, 0 \leq r_T^X, \; 0 \leq s_T^X \tag{4}$$

The variables r_T^X and s_T^X are the positive and negative feedback from a peer or collective X about peer T. It represents the trust X has in T and depends on the subjective view of X. The abbreviated notation of this formula is φ_T^X.

To communicate a reputation rating to other users the Reputation Rating function $Rep\left(r_T^X, s_T^X\right)$ is used. The function in this example gives a user a rating between $[-1; 1]$ where 0 is a neutral rating.

$$Rep(r_T^X, s_T^X) = \frac{r_T^X - s_T^X}{r_T^X + s_T^X + 2} \tag{5}$$

The formula denotes the reputation rating of T by peer or collective X. To combine feedback from different sources all positive and negative feedback is added together and the Reputation Rating is calculated. The notation of positive and negative feedback for T by peer or collective X and Y is:

$$\begin{aligned} r_T^{X,Y} &= r_T^X + r_T^Y \\ s_T^{X,Y} &= s_T^X + s_T^Y \end{aligned} \tag{6}$$

5.6 Reputation Discounting

The main idea of Reputation Discounting is that feedback given by peers with high reputation should weight more than feedback given by peers with low reputation. This idea was first described by Shafer [29] in 1976 and proposed by Jøsang [28] for usage in Trust and Reputation Systems in 2001. Applying this idea to the Beta Reputation System the following discount functions are:

$$\begin{aligned} r_T^{X:Y} &= \frac{2r_Y^X r_T^Y}{(s_Y^X + 2)(r_T^Y + s_T^Y + 2) + 2r_Y^X} \\ s_T^{X:Y} &= \frac{2r_Y^X s_T^Y}{(s_Y^X + 2)(r_T^Y + s_T^Y + 2) + 2r_Y^X} \end{aligned} \tag{7}$$

In this example X weights the transaction rating for T given by peer Y based on the current rating peer Y has.

5.7 Reputation Forgetting

Forgetting is the third concept Jøsang is using in the Beta Reputation system first proposed by Jøsang [30]. The idea is that old feedback is less relevant than recent feedback because the behaviour of a peer can change over time. For this reason Jøsang introduces a forgetting factor λ that can be adjusted individual. Where $\lambda = 0$ would mean that only the last given feedback counts and $\lambda = 1$ mean that the system doesn't forget any rating. To adapt this model it will be assumed that peers have provided a sequence Q that contains n feedback tuples indexed by i about for example peer T $(r_{T,i}^Q; s_{T,i}^Q)$. To calculate the reputation rating with the forgetting factor λ the following formula is used:

$$r_{T,\lambda}^{Q(i)} = r_{T,\lambda}^{Q(i-1)}\lambda + r_{T,i}^Q \; and \; s_{T,\lambda}^{Q(i)} = s_{T,\lambda}^{Q(i-1)}\lambda + s_{T,i}^Q \; ; \; 0 \leq \lambda \leq 1 \tag{8}$$

When a new feedback is given after a transaction by a peer the current score is multiplied by λ and the new is summed up.

6 Future Work

Through the introduction of TRS new problems and challenges emerged that need further research. Which will be outlined in this section.

6.1 Security Threads

By introducing TRS new security threads arise that TRS should be able to withstand. The researchers Marti & Garcia-Molina [24] complied a classification of possible attacks against trust and reputation systems. These attack techniques can be according to them:

Traitors A malicious peer behaves properly for a certain period of time to build up a good reputation to act malicious later. This technique is used when a peer gets additional privileges when the reputation increases.

Collusion This is the case when multiple malicious peers act together to cause more damage than they could acting alone.

Front Peers These malicious peers work together to increase their reputation by giving each other high ratings.

Whitewashers Peers that rejoin the system to get a new identity and thus get rid of their bad reputation they had with their old identity.

Denial of Service (DoS) Attacking the system from outside by sending as many requests as possible to attempt that the system is no longer reachable for everyone else.

A good overview of possible specific attacks can be found in the analysis of Gómez & Pérez [25].

6.2 Definition of a TBAC Standard

There is no formal standard for TBAC yet. Although there are different organization who tries to establish a standard like SECURE [31]. The goal of the SECURE project is to develop a trust-based generic decision-making framework for use in ubiquitous computing. The benefit of a standard would be interchangeability of trust and reputation scores and a certain level of guaranteed security and reliability through guidelines for implementation.

6.3 Fit for use

There have been many models for *Trust and Reputation computing* been proposed but I noticed that there is little or no research how to get initial trust values for the proposed systems. Yet, these are necessary to introduce TBAC in existing company information systems where disturbance of the daily operations have to be avoided when introducing this new system. Possible approaches could be that initial trust values are predefined based on organisation charts of the company.

6.4 More collaboration between researchers

I noticed that there is no significant cooperation between researchers. Thus many propose their own TRS. This leads to many similar systems and stalls further research. For example, the ratings between trust and reputation systems are not comparable as a result companies have additional effort if they would combine their systems.

7 Conclusion

TBAC and TRS are necessary in the future as decision support system and to manage tasks that are no longer sufficient to maintain manually like user privileges due increasing popularity of ubiquitous computing. But before TRS and TBAC gets widely adopted several challenges have to be solved outlined in 6. In case of rights management it would be also beneficial to think about hybrid systems that combine traditional access control mechanisms with TBAC because in many companies executives don't work very often with the IT systems directly but should still be able to access all resources even when they haven't had any transactions before with this information system in particular. Further research should be also conducted in areas like interoperability between different TRS systems by introducing a similar solution like an Enterprise Service Bus for TRS thus companies that are using different TRS systems can cooperate without additional administrative overhead.

References

[1] B. A. Misztal, *Trust in modern societies: The search for the bases of social order*. Polity Press Cambridge, 1996, vol. 1.

[2] N. Luhmann, H. Davis, J. Raffan, and K. Rooney, *Trust and Power: two works by Niklas Luhmann*. Wiley Chichester, 1979.

[3] D. Gambetta, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 2000, pp. 213–237, 2000.

[4] D. H. McKnight and N. L. Chervany, "The meanings of trust," 1996.

[5] N. Dimmock, "How much is," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. IEEE, 2003, pp. 281–282.

[6] O. E. Dictionary, "Oxford: Oxford university press," 1989.

[7] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

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

[9] L. Rasmusson and S. Jansson, "Simulated social control for secure internet commerce," in *Proceedings of the 1996 workshop on New security paradigms*. ACM, 1996, pp. 18–25.

[10] A. Tajeddine, A. Kayssi, A. Chehab, and H. Artail, *PATROL-F–a comprehensive reputation-based trust model with fuzzy subsystems.* Springer, 2006.

[11] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt, "Ratings in distributed systems: A bayesian approach," in *Proceedings of the Workshop on Information Technologies and Systems (WITS)*, 2001, pp. 1–7.

[12] A. Whitby, A. Jøsang, and J. Indulska, "Filtering out unfair ratings in bayesian reputation systems," in *Proc. 7th Int. Workshop on Trust in Agent Societies*, 2004.

[13] Y. Wang, V. Cahill, E. Gray, C. Harris, and L. Liao, "Bayesian network based trust management," in *Autonomic and Trusted Computing.* Springer, 2006, pp. 246–257.

[14] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web.* ACM, 2003, pp. 640–651.

[15] W. Nejdl, D. Olmedilla, and M. Winslett, "Peertrust: Automated trust negotiation for peers on the semantic web," in *Secure Data Management.* Springer, 2004, pp. 118–132.

[16] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on.* IEEE, 2000, pp. 9–pp.

[17] X. Ma, Z. Feng, C. Xu, and J. Wang, "A trust-based access control with feedback," in *Information Processing (ISIP), 2008 International Symposiums on.* IEEE, 2008, pp. 510–514.

[18] J. Bacon, K. Moody, and W. Yao, "A model of oasis role-based access control and its support for active security," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 4, pp. 492–540, 2002.

[19] F. Feng, C. Lin, D. Peng, and J. Li, "A trust and context based access control model for distributed systems," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on.* IEEE, 2008, pp. 629–634.

[20] S. Chakraborty and I. Ray, "Trustbac: integrating trust relationships into the rbac model for access control in open systems," in *Proceedings of the eleventh ACM symposium on Access control models and technologies.* ACM, 2006, pp. 49–58.

[21] J. Sabater and C. Sierra, "Review on computational trust and reputation models," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, 2005.

[22] M. Moloney and S. Weber, "A context-aware trust-based security system for ad hoc networks," in *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on.* IEEE, 2005, pp. 153–160.

[23] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on.* IEEE, 2000, pp. 2–14.

[24] S. Marti and H. Garcia-Molina, "Taxonomy of trust: Categorizing p2p reputation systems," *Computer Networks*, vol. 50, no. 4, pp. 472–484, 2006.

[25] F. G. Marmol and G. M. Pérez, "Security threats scenarios in trust and reputation models for distributed systems," *Computers & Security*, vol. 28, no. 7, pp. 545–556, 2009.

[26] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood, "The value of reputation on ebay: A controlled experiment," *Experimental Economics*, vol. 9, no. 2, pp. 79–101, 2006.

[27] A. Jsang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th bled electronic commerce conference*, 2002, pp. 41–55.

[28] A. Jøsang, "A logic for uncertain probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 03, pp. 279–311, 2001.

[29] G. Shafer, *A mathematical theory of evidence*. Princeton university press Princeton, 1976, vol. 1.

[30] A. Jøsang, *Modelling trust in information security*. NTNU, 1998.

[31] V. Cahill, E. Gray, J.-M. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English *et al.*, "Using trust for secure collaboration in uncertain environments," *Pervasive Computing, IEEE*, vol. 2, no. 3, pp. 52–61, 2003.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
86	978-3-86956-280-3	**Batch Regions**	Luise Pufahl, Andreas Meyer, Mathias Weske
85	978-3-86956-276-6	**HPI Future SOC Lab: Proceedings 2012**	Christoph Meinel, Andreas Polze, Gerhard Oswald, Rolf Strotmann, Ulrich Seibold, Bernhard Schulzki (Hrsg.)
84	978-3-86956-274-2	**Anbieter von Cloud Speicherdiensten im Überblick**	Christoph Meinel, Maxim Schnjakin, Tobias Metzke, Markus Freitag
83	978-3-86956-273-5	**Proceedings of the 7th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch (Hrsg.)
82	978-3-86956-266-7	**Extending a Java Virtual Machine to Dynamic Object-oriented Languages**	Tobias Pape, Arian Treffer, Robert Hirschfeld
81	978-3-86956-265-0	**Babelsberg: Specifying and Solving Constraints on Object Behavior**	Tim Felgentreff, Alan Borning, Robert Hirschfeld
80	978-3-86956-264-3	**openHPI: The MOOC Offer at Hasso Plattner Institute**	Christoph Meinel, Christian Willems
79	978-3-86956-259-9	**openHPI: Das MOOC-Angebot des Hasso-Plattner-Instituts**	Christoph Meinel, Christian Willems
78	978-3-86956-258-2	**Repairing Event Logs Using Stochastic Process Models**	Andreas Rogge-Solti, Ronny S. Mans, Wil M. P. van der Aalst, Mathias Weske
77	978-3-86956-257-5	**Business Process Architectures with Multiplicities: Transformation and Correctness**	Rami-Habib Eid-Sabbagh, Marcin Hewelt, Mathias Weske
76	978-3-86956-256-8	**Proceedings of the 6th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
75	978-3-86956-246-9	**Modeling and Verifying Dynamic Evolving Service-Oriented Architectures**	Holger Giese, Basil Becker
74	978-3-86956-245-2	**Modeling and Enacting Complex Data Dependencies in Business Processes**	Andreas Meyer, Luise Pufahl, Dirk Fahland, Mathias Weske
73	978-3-86956-241-4	**Enriching Raw Events to Enable Process Intelligence**	Nico Herzberg, Mathias Weske
72	978-3-86956-232-2	**Explorative Authoring of ActiveWeb Content in a Mobile Environment**	Conrad Calmez, Hubert Hesse, Benjamin Siegmund, Sebastian Stamm, Astrid Thomschke, Robert Hirschfeld, Dan Ingalls, Jens Lincke
71	978-3-86956-231-5	**Vereinfachung der Entwicklung von Geschäftsanwendungen durch Konsolidierung von Programmier-konzepten und -technologien**	Lenoi Berov, Johannes Henning, Toni Mattis, Patrick Rein, Robin Schreiber, Eric Seckler, Bastian Steinert, Robert Hirschfeld

ISBN 978-3-86956-281-0
ISSN 1613-5652