Design und Analyse kryptografischer Bausteine auf nicht-abelschen Gruppen

Dissertation zur Erlangung des Grades
„Doktor der Naturwissenschaften“

eingereicht beim
Fachbereich Mathematik und Informatik, Physik, Geographie
Justus-Liebig-Universität Gießen

von
Christian Tobias
aus Frankfurt am Main

Gutachter: Prof. Dr. A. Beutelspacher
Prof. Dr. J. Schwenk

Gießen, April 2004

D 26

Dekan:	Prof. Dr. V. Metag
1. Berichterstatter:	Prof. Dr. A. Beutelspacher
2. Berichterstatter:	Prof. Dr. J. Schwenk
Tag der mündlichen Prüfung:	

English Abstract

Public-key cryptography enables two (or more) parties to execute cryptographic protocols (e.g. to send confidential messages) without requiring a common secret. The methods in use today are mostly based on abelian groups (see [35] for example). Recently, several cryptographic protocols using non-abelian groups were proposed [23, 42]. The underlying security assumption on which these are based is the hardness of the conjugacy problem (or some variant). This thesis discusses the applicability of the conjugacy problem and its variants for the construction of cryptographic protocols. It is organised into two parts. In the first part the public-key cryptosystem using finite non-abelian groups proposed by Paeng et al. [42, 43] is analysed. The second part concentrates on the idea of combining the conjugacy problem (CP) and the discrete logarithm problem (DLP) to obtain new hard problems that can be used to design cryptosystems on non-abelian groups.

In 2001 Paeng et al. proposed a new public-key encryption scheme based on the difficulty of the discrete logarithm problem in the group $Inn(G)$ of inner automorphisms of a non-abelian group G [42]. Later this system was named the MOR cryptosystem [43]. The non-abelian group G to be employed has to be chosen very carefully so as not to undermine the security of the system. The first proposal for G was the semi-direct product group $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ (see [42]). The authors themselves showed an interrelation between MOR using $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ and MOR using $SL(2,\mathbb{Z}_p)$. Our security analysis of MOR using $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ reveals the following weaknesses:

- Since the conjugacy and the special conjugacy problems are easy in $SL(2,\mathbb{Z}_p)$ an attacker is able to extract the two values $\pm x\theta_1(y), \pm(x\theta_1(y))^a \in SL(2,\mathbb{Z}_p)$ from the public key. Choosing the parameters as proposed in [42] the DLP is easy in $< x\theta_1(y) >$ and the secret key a can be calculated efficiently.
  We demonstrate how to choose the parameters such that the DLP in $< x\theta_1(y) >$ is at least as hard as the DLP in subgroups of prime order of $\mathbb{Z}_p^*$.

- We present a ciphertext-only attack that can be used to calculate the encrypted plaintext message partially. The attack can be prevented by using the probabilistic padding scheme proposed in [42].

- MOR in its basic form is very inefficient (about 32 times slower than RSA). Using the probabilistic padding scheme described in [42] allows to reuse time-consuming calculations for multiple encryptions.
  We demonstrate that this variant is extremely vulnerable to known-plaintext attacks. Knowledge of two pairs consisting of plaintext and corresponding ciphertext is sufficient to efficiently compute information equivalent to the secret key.

The results of our analysis of MOR using $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ were published in [54].

In [43] Paeng et al. describe the construction of a semi-direct product group $GL(2,R) \times_\theta \mathbb{Z}_n$ from a given ring isomorphism Φ and propose to use this group for the MOR cryptosystem.

Our security analysis shows that the secret key can be (partly) calculated if the discrete logarithm problem is easy in $<\Phi>$. We further show that the efficient variants of MOR using $GL(2,R) \times_\theta \mathbb{Z}_n$ are vulnerable to chosen-plaintext attacks if the computational Diffie-Hellman problem is easy in $<\Phi>$. Our results were published in [55].

At present no group is known that can be used with the MOR system. For all suggested groups the MOR system could be shown to be insecure. We give a list of necessary conditions that a group has to fulfill to be usable with the MOR system and discuss the applicability of p-groups for MOR.

The discrete logarithm problem in $Inn(G)$ is closely related to the special conjugacy problem (SCP) and the discrete logarithm problem (DLP) in G. If G has a small center (size polynomial in the security parameter) and SCP and DLP are easy in G, then the DLP in $Inn(G)$ can also be efficiently solved. We investigate how the conjugacy problem can be combined with classical cryptographic problems such as the discrete logarithm problem or the Diffie-Hellman problems to obtain new hard problems on non-abelian groups. We give an overview of such combinations and explore relations between these problems.
The listed problems are partly used in recently published cryptographic protocols. We give a brief overview of these protocols and the level of security they offer. Some of these protocols have similar weaknesses. We point out these weaknesses and explain how they can be avoided.
In a concluding section we use one of the presented combinations of the conjugacy problem and the discrete logarithm problem to define a one-way function. We show how this function can be used to construct a simple commitment scheme which is provably secure. To demonstrate that our construction is practicable we give a realisation of the commitment scheme on $GL(2,\mathbb{Z}_p)$.

Inhaltsverzeichnis

1 Einleitung

Die Public-Key Kryptografie ermöglicht es Teilnehmern, kryptografische Protokolle auszuführen (z.B. vertrauliche Nachrichten zu versenden), ohne dass sie dazu vorher über ein gemeinsames Geheimnis verfügen müssen. Die Sicherheit bekannter Public-Key Verfahren kann nur unter der Annahme bewiesen werden, dass bestimmte mathematische Probleme nicht effizient lösbar sind. Bei diesen Annahmen handelt es sich meist um mathematische Probleme, an deren Lösung bereits von einer Vielzahl von Wissenschaftlern weltweit über einen langen Zeitraum geforscht wird, für die aber bisher trotz aller Anstrengungen keine effiziente Lösungsmethode gefunden werden konnte. Ein Beispiel dafür ist das Faktorisierungsproblem, das in der Zahlentheorie schon seit Jahrhunderten untersucht wird, jedoch immer noch als sehr schwierig gilt. In den letzten Jahrzehnten wurden zwar (insbesondere durch den Einsatz moderner Computertechnik) immer effizientere Verfahren zur Faktorisierung entwickelt. Die erzielten Ergebnisse reichen aber bei weitem noch nicht aus, um die modernen Public-Key Verfahren ernsthaft angreifen zu können.
Es ist zwar unwahrscheinlich aber durchaus denkbar, dass durch neue Ergebnisse in der Mathematik oder Informatik so große Fortschritte bei der Lösung einiger dieser Probleme gemacht werden, dass sich einzelne Kryptoverfahren oder sogar ganze Klassen von Kryptoverfahren als unsicher erweisen. Um für einen solchen Fall gerüstet zu sein, versucht man neue Public-Key Verfahren zu entwickeln, die auf anderen Annahmen basieren. Sollte sich ein Durchbruch bei der Lösung eines für die Kryptografie grundlegenden Problems abzeichnen, können die eingesetzten Systeme rechtzeitig auf neue Verfahren umgestellt werden, deren Sicherheit nicht durch die neuen Entdeckungen gefährdet sind.

Die heutzutage gebräuchlichen Public-Key Verfahren basieren auf endlichen abelschen Gruppen (siehe etwa [4, 35]). In letzter Zeit werden erhebliche Anstrengungen unternommen, kryptografische Verfahren auf nicht-abelschen Gruppen zu konstruieren. Dabei wird die Schwierigkeit des Konjugationsproblems (CP, Conjugacy Problem) oder einer Variante als zugrundeliegende Sicherheitsannahme verwendet, d.h. man nimmt an, dass es in einer Gruppe G keinen effizienten Algorithmus gibt, der für gegebene $x, y \in G$ mit $y = gxg^{-1}$ für ein $g \in G$ ein $\bar{g} \in G$ berechnen kann mit $y = \bar{g}x\bar{g}^{-1}$.
In Matrixgruppen ist das Konjugationsproblem effizient lösbar. Dazu betrachtet man die zur Gleichung $y = gxg^{-1}$ äquivalente Darstellung $yg = gx$. Das Lösen des Konjugationsproblems in der Matrixgruppe reduziert sich so auf das Lösen eines linearen Gleichungssystems über der der Matrixgruppe zugrundeliegenden Struktur (etwa einem Körper oder einem Ring). Da die Effizienz von Berechnungen in der Kryptografie eine wesentliche Rolle spielt, werden oft sehr kleine Matrizen (mit wenigen Einträgen) verwendet. Das einer Instanz des Konjugationsproblems zugehörige lineare Gleichungssystem besteht also meist nur aus wenigen Gleichungen und kann effizient gelöst werden.
Durch die Verknüpfung des Konjugationsproblems mit dem Diskreter Logarithmus Problem (DLP, discrete logarithm problem) wird das Konjugationsproblem auf Matrixgruppen dennoch für die Kryptografie interessant. Das Diskreter Logarithmus Problem ist eines der in der Kryptografie etablierten Probleme. Dabei geht man von einer zyklischen Gruppe G

und einem Generator g von G aus. Für ein beliebiges $h \in G$ besteht das DLP darin, die eindeutig bestimmte ganze Zahl $0 \leq x \leq \text{ord}(G) - 1$ mit $g^x = h$ zu finden. Man nimmt an, dass das DLP in $\mathbb{Z}_p^*$ (p prim) und Untergruppen primer Ordnung von $\mathbb{Z}_p^*$ schwer ist. Das DLP lässt sich auf Matrixgruppen übertragen, etwa auf zyklische Untergruppen der Gruppe $GL(2, \mathbb{Z}_p)$ der invertierbaren 2×2 Matrizen über dem Körper $\mathbb{Z}_p$. Die Gruppe $GL(2, \mathbb{Z}_p)$ hat die Ordnung $(p-1)^2(p+1)p$. Damit definiert jedes Element aus $GL(2, \mathbb{Z}_p)$ eine endliche zyklische Untergruppe, in der man das DLP betrachten kann.
Kombiniert man nun das Konjugationsproblem mit dem DLP, so erhält man das folgende Problem, das wir *Diskreter Logarithmus Konjugationsproblem* (DLCP, discrete logarithm conjugacy problem) nennen werden:

> Gegeben seien eine nicht-abelsche Gruppe G sowie Elemente $g, x, y \in G$ mit $y = g^a x g^{-a}$. Gesucht ist a.[1]

Die für die Kryptografie interessante Tatsache ist dabei, dass ein Angreifer, der das CP und das DLP in der benutzten nicht-abelschen Gruppe G lösen kann, nicht unbedingt in der Lage ist, das DLCP in G zu lösen. Das liegt daran, dass das CP im Allgemeinen keine eindeutige Lösung hat. Die Anzahl der Lösungen des CP hängt von der Größe des Zentralisators $Z(x)$ des konjugierten Elements x ab. Das Element $\bar{g} \in G$ ist genau dann eine Lösung der Instanz x, gxg^{-1} des CP in G, d.h. es gilt $gxg^{-1} = \bar{g}x\bar{g}^{-1}$, wenn $\bar{g}$ von der Form $\bar{g} = x \cdot z$ mit $z \in Z(x)$ ist.
Es sei mit $g, x, g^a x g^{-a}$ eine Instanz des DLCP gegeben. Dabei seien $g, x \in G$ so gewählt, dass $g \notin Z(x)$ und $\text{ord}(g) = p$ mit p prim ist. Durch Lösen der Instanz $x, g^a x g^{-a}$ des CP erhält ein Angreifer einen Wert $\hat{g}$ mit $g^a x g^{-a} = \hat{g} x \hat{g}^{-1}$ und $\hat{g} = g^a \cdot z$ für ein $z \in Z(x)$. Dabei liegt $\hat{g}$ genau dann in $< g >$, wenn $z \in < g >$ gilt. Aufgrund der speziellen Wahl von g und x kann dies nur erfüllt sein, wenn $z = 1_G$. Unter allen $|Z(x)|$ Lösungen der Instanz $x, g^a x g^{-a}$ des CP gibt es also genau eine Lösung $\hat{g}$, so dass $g, \hat{g}$ eine Instanz des Diskreter Logarithmus Problems in G ist. Bevor der Angreifer seinen Algorithmus zur Lösung des DLP in G einsetzen kann, muss er zunächst die Lösung des CP „herausfiltern“, die zu einer Instanz des DLP führt. Wählt man nun eine nicht-abelsche Gruppe G mit einem hinreichend großen Zentrum, so kann der Angreifer eine solche Instanz des DLP durch Ausprobieren nicht effizient finden. Die Größe des Zentrums ist allerdings nur ein notwendiges aber noch kein hinreichendes Kriterium für die Schwierigkeit des DLCP in G. Ein Beispiel für eine Gruppe mit einem großen Zentrum, in der das DLCP äquivalent zum DLP ist, ist die lineare Gruppe $GL(2, \mathbb{Z}_p)$.

Eng verbunden mit dem DLCP auf nicht-abelschen Gruppe G ist das DLP in der Gruppe $Inn(G)$ der inneren Automorphismen von G. Jedes g definiert durch $Inn(g) : G \to G$, $x \mapsto gxg^{-1}$ einen (inneren) Automorphismus. Die Menge $Inn(G) = \{Inn(g) \mid g \in G\}$ aller solcher Automorphismen bildet bzgl. der Hintereinanderausführung eine Gruppe, die

[1] Damit dieses Problem eine eindeutige Lösung hat, müssen eventuell Einschränkungen an die Wahl der Parameter gemacht werden, siehe Kapitel 7.

Gruppe der inneren Automorphismen von G. Sei mit $Inn(g), Inn(g^a) = (Inn(g))^a$ eine Instanz des DLP in $Inn(G)$ gegeben. Wir nehmen dabei an, dass Funktionswerte von $Inn(g)$ und $Inn(g^a)$ berechnet werden können, die definierenden Elemente g und g^a selbst aber nicht bekannt sind.[2] Beim speziellen Konjugationsproblem (special conjugacy problem, SCP) muss für einen gegebenen inneren Automorphismus $\varphi \in Inn(G)$ ein definierendes Element gefunden werden, d.h. ein $g \in G$ mit $Inn(g) = \varphi$. Genau wie das CP hat auch das SCP keine eindeutige Lösung: es gilt $Inn(g) = Inn(\bar{g})$ genau dann ,wenn $\bar{g}$ von der Form $\bar{g} = g \cdot z$ für ein Element $z \in Z(G)$ aus dem Zentrum $Z(G)$ von G ist. Ein Angreifer, der das SCP und das DLP in G lösen kann, ist wieder nicht unbedingt in der Lage, das DLP in $Inn(G)$ zu lösen. Die Größe des Zentrums $Z(G)$ der benutzten Gruppe G ist ein notwendiges (aber kein hinreichendes) Kriterium für die Schwierigkeit des DLP in $Inn(G)$. Auf der Crypto 2001 [42] wurde mit dem MOR Verfahren ein asymmetrisches Verschlüsselungsverfahren vorgestellt, dessen Sicherheit auf der Schwierigkeit des DLP in $Inn(G)$ beruht. Bei diesem Verfahren besteht der öffentliche Schlüssel eines Teilnehmers aus den beiden Funktionen $Inn(g)$ und $Inn(g^a)$. Der diskrete Logarithmus a bildet den geheimen Schlüssel. Ein Angreifer, der diskrete Logarithmen in $Inn(G)$ berechnen kann, ist damit in der Lage, den geheimen Schlüssel berechnen. Die Gruppe G muss demnach so gewählt werden, dass das DLP in $Inn(G)$ schwer ist.
In [42] wird kein Beweis für die Sicherheit des MOR Systems geführt. Es werden lediglich notwendige (aber keine hinreichenden) Bedingungen für die Sicherheit des Systems angegeben. Tatsächlich ist in den in [42, 43] vorgeschlagenen Gruppen das Berechnen des geheimen Schlüssels a nicht nötig, um Klartexte (oder Teile davon) effizient rekonstruieren zu können (siehe [54, 55] und Kapitel 4 und 5). Die vorliegende Arbeit ist zu einem großen Teil von dem von Paeng, Ha, Kim, Chee und Park verfassten Artikel über das MOR Kryptosystem [42] motiviert.

1.1 Frühere Arbeiten

Eines der ersten Kryptosysteme, das Ergebnisse der kombinatorischen Gruppentheorie in der Kryptografie anwendet und eine Operation der Gruppe $SL(2, \mathbb{Z})$ auf der Menge $\{z \in \mathbb{C} \mid Im(z) > 0\}$ benutzt, wurde von Yamamura im Jahre 1998 vorgestellt [59]. Ein Jahr später hat Yamamura eine verbesserte Version [60] vorgestellt. Im selben Jahr haben Blackburn und Galbraith einen Angriff vorgestellt, mit dem für beide Versionen die Berechnung von Klartexten aus den zugehörigen Geheimtexten effizient möglich ist [5].

Bei dem im Jahre 2001 vorgestellten MOR Kryptosystem [42] handelt es sich um eine Variante der ElGamal Verschlüsselung auf einer zyklischen Untergruppe der Gruppe der inneren Automorphismen einer nicht-abelschen Gruppe. Für die zunächst vorgeschlagene Gruppe $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ hat Tobias [54] Angriffe vorgestellt, die im Extremfall die Berechnung eines zum geheimen Schlüssel äquivalenten Wertes ermöglichen. Auch für die später

[2] Bei endlich erzeugten Gruppen G mit Generatoren $g_1, \ldots, g_n$ ist die Funktion $Inn(g)$ bereits durch die Bilder $Inn(g)(g_1), \ldots, Inn(g)(g_n)$ eindeutig bestimmt.

in [43] vorgeschlagene Gruppe $GL(2, R) \times_\theta \mathbb{Z}_n$ gibt es Angriffe, mit denen Informationen über die verschlüsselte Klartextnachricht berechnet werden können (siehe [55] und Kapitel 5). Da das MOR System einen Schwerpunkt der vorliegenden Arbeit bildet, werden das MOR System selbst, die vorgeschlagenen Gruppen sowie deren Analyse in späteren Kapiteln dieser Arbeit ausführlich behandelt.

In [2] und [1] stellen Anshel, Anshel, Goldfeld und Fisher ein Protokoll zur Schlüsselvereinbarung auf nicht-abelschen Gruppen vor, in denen das Multiple Conjugator Search Problem (MCSP) schwer ist. Bei dem vorgestellten Protokoll handelt es sich um eine Variante des Diffie-Hellman Protokolls [13]. Die Idee des Protokolls besteht darin, jedem Teilnehmer eine endlich erzeugte Untergruppe der benutzten nicht-abelschen Gruppe G zuzuordnen. Die Erzeuger dieser Untergruppe sind öffentlich. Mittels der Homomorphie-Eigenschaft der Konjugationsabbildung können sich zwei Parteien auf ein gemeinsames Geheimnis einigen.

Im Jahr 2000 wurden erstmals kryptografische Verfahren auf Zopfgruppen vorgestellt [23]. Es wurden ein Protokoll zur Schlüsselvereinbarung, das dem Diffie-Hellman Protokoll ähnlich ist, und ein asymmetrisches Verschlüsselungsverfahren, das dem ElGamal Verfahren ähnlich ist, vorgeschlagen.
Als zugrundeliegende Sicherheitsannahme verwenden diese Verfahren die Schwierigkeit des Diffie-Hellman Konjugationsproblems. Bisher sind keine Algorithmen bekannt, die das Konjugationsproblem oder das Diffie-Hellman Konjugationsproblem für den allgemeinen Fall auf Zopfgruppen effizient lösen. Für die Instanzen, die in [23] vorkommen, kennt man jedoch effiziente Lösungsmethoden. Hofheinz und Steinwandt [19] geben einen heuristischen Algorithmus für das Konjugationsproblem an, der auf Vorarbeiten von Hughes [20] sowie Lee und Lee [30] basiert. Der Algorithmus nutzt spezielle Eigenschaften der Schlüsselerzeugung aus. In [8] wird eine Weiterentwicklung dieser Angriffe vorgestellt, die auch für die in [28] vorgestellten Varianten der Protokolle aus [23] anwendbar ist. In [52] werden Verbesserungen für die Schlüsselerzeugung vorgeschlagen, die den Angriff unwirksam werden lassen. Im letzten Jahr haben Cheon und Jun [10] einen Algorithmus mit polynomieller Laufzeit für die in der Kryptografie auf Zopfgruppen auftretenden Instanzen des Diffie-Hellman Konjugationsproblems vorgestellt. Die Laufzeit dieses Algorithmus beträgt $\mathcal{O}(n^{14.4}l^{3.2})$, wobei n der Index der benutzten Zopfgruppe und l die Charney Länge der benutzten Zöpfe ist. Die Laufzeit des Algorithmus ist zwar noch zu hoch, um Instanzen des Diffie-Hellman Konjugationsproblems mit den derzeitig für die Kryptografie auf Zopfgruppen üblichen Parametern anzugreifen. Die Existenz eines Algorithmus mit polynomieller Laufzeit und die offene Frage, in wie weit sich die Effizienz des Algorithmus noch steigern lässt, mindern jedoch das Vertrauen in diese Protokolle erheblich.

Die in [2, 1] vorgestellten Protokolle zur Schlüsselvereinbarung auf nicht-abelschen Gruppen, in denen das Multiple Conjugator Search Problem (MCSP) schwer ist, lassen sich auf Zopfgruppen übertragen. Die in [19, 20, 30] vorgestellten Angriffe funktionieren auch für Instanzen des Konjugationsproblems, wie sie in [2, 1] auftreten. Derzeit gibt es keine Vorschläge, wie diese Protokolle gegen die erwähnten Angriffe abgesichert werden können.

1.2 Aufbau und Ergebnisse der Arbeit

In Kapitel 2 werden zunächst Notationen, Begriffe und Konstruktionen eingeführt, die im weiteren Verlauf Verwendung finden. Kapitel 3 enthält eine Vorstellung des MOR Kryptosystem in seiner Grundversion [42] sowie in einer erweiterten Version [43]. Es werden die Effizienz des MOR Kryptosystems analysiert und notwendige Bedingungen für die Wahl einer geeigneten nicht-abelschen Gruppe formuliert. Es folgt eine detaillierte Analyse des MOR Systems auf den in [42] und [43] vorgeschlagenen Gruppen in den Kapiteln 4 und 5.

Der erste Vorschlag einer Gruppe für das MOR System war das semi-direkte Produkt $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ [42]. Die Analyse von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ bildet einen Schwerpunkt von Kapitel 4 sowie der vorliegenden Arbeit. Dabei wurden unter anderem folgende Schwächen des Verfahrens gefunden:

- Durch die spezielle Wahl des Homomorphismus θ als Komposition eines Homomorphismus θ_1 und eines inneren Automorphismus kann MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ auf MOR auf $SL(2,\mathbb{Z}_p)$ reduziert werden. Wegen der Invarianz von Spur und Determinante in $GL(2,\mathbb{Z}_p)$ unter Konjugation, bildet diese Reduktion die Basis für eine Reihe von schwerwiegenden Angriffen.

- Da das Konjugationsproblem in $SL(2,\mathbb{Z}_p)$ leicht ist, kann ein Angreifer bei MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ aus dem öffentlichen Schlüssel $Inn(g)$, $Inn(g^a)$, wobei $g = (x,y) \in SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ ist, die Werte $x\theta_1(y)$ und $(x\theta_1(y))^a$ berechnen.
  In [42] wurden die Parameter und damit insbesondere auch g so gewählt, dass effizient ver- und entschlüsselt werden kann. Allerdings ist für die vorgeschlagene Wahl von g die Berechnung diskreter Logarithmen in $< x\theta_1(y) >$ leicht, so dass aus $x\theta_1(y)$ und $(x\theta_1(y))^a$ der geheime Schlüssel a effizient berechnet werden kann.
  In Kapitel 4 wird gezeigt, wie man die Wahl der Parameter ändern muss, damit das Diskreter Logarithmus Problem in $< x\theta_1(y) >$ mindestens so schwer ist wie in $\mathbb{Z}_p^*$.

- Mittels eines Ciphertext-Only Angriffes kann die Struktur der verschlüsselten Nachricht extrahiert werden. Dadurch kann für einen gegebenen Geheimtext die Menge der möglichen Klartexte auf eine exponentiell kleine Teilmenge des Klartextraumes eingeschränkt werden. Dieser Angriff kann durch Benutzung des in [42] vorgeschlagenen Padding-Verfahrens verhindert werden.

- MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ ist in seiner Grundform über 32-mal langsamer als RSA bei vergleichbarer Schlüssellänge (siehe Abschnitt 4.1) und damit zu rechenaufwändig für die meisten Anwendungen. Aus diesem Grund schlagen die Autoren in [42] ein probabilistisches Padding-Verfahren vor, um rechenintensive Schritte aus Ver- und Entschlüsselung für mehrere Klartexte nutzen zu können. In Abschnitt 4.3 wird gezeigt, dass bereits zwei Paare bestehend aus Klartext und zugehörigem Geheimtext ausreichen, um alle folgenden Geheimtexte effizient entschlüsseln zu können.

Die Ergebnisse der Analyse von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ wurden im Januar 2003 auf dem Workshop für Public-Key Cryptography, PKC 2003, vorgestellt [54]. Derzeit gibt es keine

Vorschläge, wie man MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ abändern könnte, um die beschriebenen Angriffe zu verhindern (z.B. durch ein verbessertes Padding).

In [43] wird ein Konstruktionsprinzip vorgestellt, mittels dem aus einem gegebenen Ringautomorphismus Φ eine Gruppe $GL(2, R) \times_\theta \mathbb{Z}_n$ abgeleitet werden kann. Wir stellen diese Konstruktion in Kapitel 5 vor und diskutieren ihre Nutzbarkeit für das MOR System. Unsere Analyse zeigt, dass die Schwierigkeit des Diskreter Logarithmus Problems und des Computational Diffie-Hellman Problems in der von Φ erzeugten Gruppe $<\Phi>$ grundlegend für die Sicherheit von MOR auf $GL(2, R) \times_\theta \mathbb{Z}_n$ ist. In allen Betriebsmodi lassen sich Informationen über den geheimen Schlüssel berechnen, wenn diskrete Logarithmen in $<\Phi>$ effizient berechnet werden können. In einigen Modi können durch einen Chosen-Ciphertext Angriff Geheimtexte vollständig entschlüsselt werden, wenn das Computational Diffie-Hellman Problem in $<\Phi>$ effizient gelöst werden kann.
Da die Schwierigkeit des DLP in $<\Phi>$ eine wesentlich schwächere Annahme ist als die Schwierigkeit des DLP in der inneren Automorphismengruppe von G, schwächen die vorgestellten Angriffe das MOR System deutlich. Bei allen vorgestellten Angriffen handelt es sich um generische Angriffe, die auf jedem Ring R durchführbar sind. Bei der Wahl eines konkreten Rings kann es aber durchaus noch stärkere Angriffe geben.
Die Ergebnisse der Sicherheitsanalyse von MOR auf $GL(2, R) \times_\theta \mathbb{Z}_n$ werden im April 2004 im Rahmen des Second International Workshop on Security in Information Systems, WOSIS 2004, vorgestellt und veröffentlicht werden (siehe [55]).

Die Ergebnisse dieser Arbeit zeigen, dass MOR auf den in [42, 43] vorgeschlagenen Gruppen erhebliche Sicherheitslücken aufweist. Derzeit ist keine Gruppe bekannt, auf der das MOR System sicher ausgeführt werden kann. Kapitel 6 beschäftigt sich mit der Eignung von p-Gruppen für das MOR System. Zunächst werden notwendige Anforderungen an Gruppen für ihre Benutzbarkeit mit dem MOR System formuliert. Wir geben eine Darstellung von p-Gruppen an, die eine effiziente Implementierbarkeit und eine effiziente Lösung des Wortproblems ermöglicht. Unter der Annahme, dass das DLP schwer ist in $Inn(G)$, erfüllen p-Gruppen die notwendigen Bedingungen für eine Benutzung im MOR System.

Kapitel 7 beschäftigt sich mit kryptografischen Problemen, die eine Kombination des Konjugationsproblems und des Diskreter Logarithmus Problems darstellen.
In Abschnitt 7.1 werden zunächst bekannte und neue Kombinationen vorgestellt und diskutiert. Ein Schwerpunkt der Analyse bildet dabei die Frage, in wie weit die vorgestellten Probleme voneinander abhängen (d.h. sich aufeinander reduzieren lassen) und wie schwer diese Probleme in $GL(2, \mathbb{Z}_p)$ sind. Auf Grundlage dieser Probleme wird anschließend eine neue Einwegfunktion angegeben.
In Abschnitt 7.2 werden mit den in [23] vorgestellten Algorithmen zum Schlüsselaustausch und zur Verschlüsselung auf Zopfgruppen und dem Cayley-Purser-Algorithmus [16] kryptografische Verfahren vorgestellt, deren Sicherheit auf den in Abschnitt 7.1 behandelten Problemen beruht.
Die in [23] vorgestellten Verfahren lassen sich auf andere nicht-abelsche Gruppen über-

tragen. Benutzt man die Gruppe $GL(2, \mathbb{Z}_p)$ anstatt der vorgeschlagenen Zopfgruppen, so ist es jedoch möglich, den geheimen Schlüssel aus dem öffentlichen Schlüssel zu berechnen. Beim Cayley-Purser-Algorithmus, der auf $GL(2, \mathbb{Z}_n)$ arbeitet, ist dies ebenfalls möglich. Ein Vergleich, der in [23] vorgestellten Algorithmen mit dem MOR Kryptosystem und dem Cayley-Purser-Algorithmus ergibt, dass alle Verfahren über die selbe Konstruktionsschwäche verfügen, die ein Angreifer ausnutzen kann. Die Ursachen dieser Konstruktionsschwäche werden anschließend erörtert und ein Konstruktionsprinzip aufgestellt, das helfen soll, diesen Fehler im Design von kryptografischen Verfahren auf nicht-abelschen Gruppen in Zukunft zu vermeiden.
Abschnitt 7.3 baut auf der in Abschnitt 7.1 vorgestellten Einwegfunktion auf, um ein beweisbar sicheres Commitment Scheme zu konstruieren. Das Commitment Scheme wird zunächst für beliebige nicht-abelsche Gruppen vorgestellt. Ferner werden hinreichende Bedingungen für dessen Sicherheit formuliert. Anschließend wird eine Realisierung des Verfahrens auf der Gruppe $GL(2, \mathbb{Z}_p)$ angegeben. Die konkrete Umsetzung zeigt zum einen, dass es möglich ist, auf $GL(2, \mathbb{Z}_p)$ kryptografische Verfahren zu konstruieren, die beweisbar sicher sind. Zum anderen demonstriert sie, dass die angegebenen Bedingungen für die Sicherheit des Commitment Schemes in speziellen Gruppen tatsächlich umsetzbar sind.

Mein Dank gilt Herrn Prof. Dr. Albrecht Beutelspacher für die stets kompetente und engagierte Betreuung dieser Arbeit, Herrn Prof. Dr. Reinhard Laue für die fruchtbare Unterstützung bei der Suche nach neuen Gruppen für das MOR System und deren Implementierbarkeit, Herrn Prof. Dr. Bernd Baumann für hilfreiche Diskussionen über Gruppentheorie, Herrn Prof. Dr. Jörg Schwenk für die Bereitschaft, diese Arbeit zu begutachten, Herrn Prof. Dr. Claus-Peter Schnorr für Unterstützung bei der Wahl des Themas, Matthias Baumgart und Dennis Hofheinz für interessante Diskussionen über Matrix- bzw. Zopfgruppen und Christine Fremdt, Dr. Heike Neumann, Björn Fay, Thomas Schwarzpaul und Jörn Schweisgut für wertvolle Verbesserungsvorschläge.
Darüber hinaus möchte ich mich bei meiner Frau Daniela, meiner Familie sowie Freunden und Kollegen für ihre Unterstützung und Aufmunterung bedanken.

2 Grundlagen

Der nun folgende Abschnitt dient dazu, den theoretischen Unterbau für das Verständnis der späteren Kapitel bereitzustellen. Dabei wird vorausgesetzt, dass der Leser mit den grundlegenden Konzepten und der Funktionsweise der asymmetrischen Kryptografie vertraut ist. Abschnitt 2.1 führt zunächst einige der im Weiteren benutzten Notationen ein. In den Abschnitten 2.2 bis 2.5 wird eine kurze Einführung in die Komplexitätstheorie und ihre Anwendung in der Kryptografie gegeben. Ferner werden das Diskreter Logarithmus Problem und die Diffie-Hellman Probleme diskutiert und gezeigt, wie diese im Diffie-Hellman Protokoll zur Schlüsselvereinbarung und im ElGamal Verschlüsselungsverfahren eingesetzt werden.
Die Abschnitte 2.6 bis 2.9 stellen die zur Realisierung kryptografischer Verfahren auf nichtabelschen Gruppen benötigten Grundlagen vor. Nach der Einführung der Konjugationsprobleme, deren Schwierigkeit als zugrundeliegende Sicherheitsannahme für die betrachteten Verfahren dient, werden die linearen Gruppen $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$ sowie das semidirekte Produkt zweier Gruppen eingeführt. Der Fokus der Betrachtungen liegt dabei auf den Eigenschaften dieser Gruppen, die eine Bedeutung für die in dieser Arbeit betrachteten kryptografischen Verfahren haben.

2.1 Notationen

Im Laufe dieser Arbeit werden folgende Notationen verwendet, die zwar in der Kryptografie üblich sind, in anderen Gebieten der Mathematik aber teils ungewöhnlich erscheinen mögen:

- **Wahl von Zufallselementen:** Die Notation $g \in_R G$ wird immer dann verwendet, wenn ein Element g gemäß der Gleichverteilung auf G zufällig gewählt wird.
- **Verwendung von Restklassen:** An einigen Stellen verwenden wir $\mathbb{Z}_p = \mathbb{Z}/p\,\mathbb{Z}$ nicht als das System der Restklassen der ganzen Zahlen modulo p, sondern als das kleinste nichtnegative Restsystem $0, 1, \ldots, p-1$ modulo p. Wir identifizieren die Restklasse $[x] \in \mathbb{Z}_p$ in diesen Fällen mit ihrem kleinsten nichtnegativen Repräsentanten $(x \bmod p)$ und rechnen mit ihm wie mit ganzen Zahlen.
- **Das Symbol δ_{ij}:** Das Symbol δ_{ij} wird in dieser Arbeit nicht zur Bezeichnung des Kronecker-Symbols benutzt, sondern dient als abkürzende Schreibweise für Matrizen, deren (i, j)-Komponente den Wert 1 und deren übrige Komponenten den Wert 0 haben (siehe auch Abschnitt 2.7.1 auf Seite 24).

2.2 Komplexitätstheorie

Das Hauptziel der Komplexitätstheorie ist es, Probleme nach den für ihre Lösung notwendigen Ressourcen zu klassifizieren. Die für eine solche Klassifizierung betrachteten Ressourcen sind meistens die Laufzeit und der Speicherverbrauch.

2.2.1 Die Komplexitätsklassen $\mathcal{P}$, $\mathcal{NP}$ und polynomielle Reduzierbarkeit

Die in dieser Arbeit betrachteten Probleme sind entweder Berechnungs- oder Entscheidungsprobleme. Bei Berechnungsproblemen soll für eine Eingabe $x \in \{0,1\}^*$ eine Ausgabe berechnet werden, die mit der Eingabe in einer bestimmten Relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ steht. Die Ausgabe eines Primteilers bei Eingabe einer natürlichen Zahl ist ein Beispiel für ein Berechnungsproblem. Bei Entscheidungsproblemen soll eine gegebene Sprache L und für eine Eingabe $x \in \{0,1\}^*$ entschieden werden, ob x in L liegt. Entscheidungsprobleme können als Spezialfall $R \subseteq \{0,1\}^* \times \{0,1\}$ von Berechnungsproblemen mit der zusätzlichen Bedingung $(x,1) \in R \Leftrightarrow (x,0) \notin R \Leftrightarrow x \in L$ aufgefasst werden.
Eine Turingmaschine M berechnet eine Funktion $f_M : \{0,1\}^* \to \{0,1\}^*$, wenn M auf Eingabe x mit Ausgabe $f_M(x)$ hält. Sie löst ein Problem $R \subseteq \{0,1\}^* \times \{0,1\}^*$, wenn für alle $(x,y) \in R$ auch $(x, f_M(x)) \in R$ gilt. Für jede Instanz x, für die es eine Lösung y gibt, d.h. $(x,y) \in R$ gilt, findet M eine Lösung $\hat{y}$, d.h. $(x,\hat{y})$.
Aufgrund der Churchschen These, die besagt, dass es genau dann ein (stets haltendes) *Rechenverfahren* zur Berechnung einer Funktion gibt, wenn diese Funktion durch eine (stets haltende) Turingmaschine berechnet werden kann, wurde die Turingmaschine als zugrundeliegendes Rechenmodell gewählt. Die Turingmaschine dient als Abstraktion echter Computer und könnte aufgrund der Churchschen These durch andere Rechenmodelle ersetzt werden. Im Folgenden werden die Begriffe *Turingmaschine* und *Algorithmus* synonym verwendet.

Als effizient lösbar bezeichnet man die Probleme, für die es einen Algorithmus gibt, dessen Laufzeit polynomiell in der Eingabelänge beschränkt ist. Polynome stellen eine Klasse von Funktionen dar, deren asymptotisches Wachstum anhand ihres höchstens Exponenten leicht kategorisiert werden kann und die über für Berechnungen nützliche Abschlusseigenschaften verfügen. Die Klasse der Polynome ist nämlich abgeschlossen gegenüber Addition, Multiplikation und Komposition. Die Probleme, die nach diesen Maßstäben effizient lösbar sind, fassen wir in der Klasse P zusammen.

Definition 1 (Komplexitätsklasse $\mathcal{P}$) *Die Komplexitätsklasse $\mathcal{P}$ ist die Menge aller Probleme, für die es eine (deterministische) Turingmaschine $\mathcal{M}$ gibt, deren worst case Laufzeit polynomiell (in der Eingabelänge) beschränkt ist.*

Wir erweitern nun unser Rechenmodell von deterministischen zu nicht-deterministischen Turingmaschinen. Bei nicht-deterministischen Turingmaschinen ergibt sich die Folgekonfiguration nicht als Funktion (wie bei deterministischen Turingmaschinen) sondern als Relation aus der aktuellen Konfiguration. Die nicht-deterministische Turingmaschine hat also zur Laufzeit die Wahl aus mindestens einer möglichen Folgekonfiguration.

Definition 2 (Komplexitätsklasse $\mathcal{NP}$) *Die Komplexitätsklasse $\mathcal{NP}$ ist die Menge aller Probleme, für die es eine nicht-deterministische Turingmaschine $\mathcal{M}$ gibt, deren worst case Laufzeit polynomiell (in der Eingabelänge) beschränkt ist.*

Die Menge $\mathcal{NP}$ aller durch nicht-deterministische Turingmaschinen lösbare Probleme besteht aus den Problemen, deren Lösung, wenn sie gegeben ist, effizient überprüft werden kann. Das Problem, zu einer gegebenen zusammengesetzten Zahl n einen Teiler m zu finden, liegt zum Beispiel in $\mathcal{NP}$. Ist m gegeben, so kann durch Division effizient überprüft werden, ob $m \mid n$ tatsächlich erfüllt ist.

Es ist oftmals der Fall, dass ein Problem durch Benutzung eines effizienten Unterprogramm für ein anderes Problem effizient gelöst werden kann. Dieses Prinzip der Reduktion eines Problems auf ein anderes ist ein wichtiges Prinzip der Komplexitätstheorie.

Definition 3 (Polynomielle Reduzierbarkeit) *Ein Problem P_1 heißt polynomiell reduzierbar auf P_2, geschrieben $P_1 \leq_P P_2$, wenn es einen deterministischen Algorithmus $\mathcal{A}$ mit polynomieller worst case Laufzeit gibt, der P_1 löst. Algorithmus $\mathcal{A}$ kann dabei auf einen Algorithmus $\bar{\mathcal{A}}$, der P_2 löst, als Unterroutine zugreifen.*

Die Aussage $P_1 \leq_P P_2$ bedeutet also, dass mit P_2 auch P_1 in polynomieller Laufzeit lösbar ist. Problem P_2 ist also mindestens so schwer ist wie P_1.

Definition 4 *Seien P_1 und P_2 zwei Probleme. Wenn $P_1 \leq_P P_2$ und $P_2 \leq_P P_1$ gilt, so nennt man P_1 und P_2 (rechnerisch) äquivalent, geschrieben $P_1 =_P P_2$.*

Das Prinzip der Reduktion wird dazu verwendet, um die Teilmenge der $\mathcal{NP}$-vollständigen Probleme von $\mathcal{NP}$ zu definieren.

Definition 5 ($\mathcal{NP}$-Vollständigkeit) *Ein Problem P heißt $\mathcal{NP}$-vollständig, wenn es in $\mathcal{NP}$ ist und $K \leq_P P$ für alle $K \in \mathcal{NP}$ gilt.*

Der nächste Satz zeigt, dass die $\mathcal{NP}$-vollständigen Probleme die schwierigsten Probleme in $\mathcal{NP}$ sind.

Satz 1 *Seien P_1, P_2, P_3 und P Probleme.*

1. *Es gelte $P_1 \leq_P P_2$ und $P_2 \in \mathcal{P}$. Dann ist auch $P_1 \in \mathcal{P}$.*
2. *Wenn P ein $\mathcal{NP}$-vollständiges Problem ist und $\mathcal{P} \neq \mathcal{NP}$ ist, dann ist $P \notin \mathcal{P}$.*
3. *Wenn $P_1 \leq_P P_2$ und $P_2 \leq_P P_3$, dann ist $P_1 \leq_P P_3$.*

Beweis: Siehe [57].

Aus dem Satz folgt ferner, dass die Klassen $\mathcal{P}$ und $\mathcal{NP}$ gleich sind, wenn für ein $\mathcal{NP}$-vollständiges Problem gezeigt werden kann, dass es in $\mathcal{P}$ liegt. Da deterministische Turingmaschinen Spezialfälle von nicht-deterministischen Turingmaschinen sind, ist direkt aus den Definitionen ersichtlich, dass $\mathcal{P} \subseteq \mathcal{NP}$ gilt. Eines der großen offenen Probleme der Komplexitätstheorie ist die Frage, ob diese Inklusion echt ist. Man vermutet, dass $\mathcal{P} \neq \mathcal{NP}$ gilt. Empirisch wird diese Vermutung durch tausende Probleme gestützt, von denen man zeigen kann, dass sie $\mathcal{NP}$-vollständig sind, für keines dieser Probleme jedoch bislang eine effiziente Lösung gefunden werden konnte.

2.2.2 Randomisierte Algorithmen

Bisher wurden nur deterministische Algorithmen betrachtet. Diese liefen bei gleicher Eingabe auch immer das gleiche Ergebnis. In der Praxis werden oft probabilistische Algorithmen benutzt, deren Berechnungen von Zufallsentscheidungen abhängen können. Solche Algorithmen werden zum Beispiel zur Durchführung von Simulationen eingesetzt.
Probabilistische Algorithmen kann man sich als Turingmaschinen vorstellen, die neben dem eigentlichen Eingabeband über ein weiteres Band mit Zufallsbits verfügen. Die durchgeführten Berechnungen hängen von beiden Bändern ab. Das Band mit den Zufallsbits wird dabei für jede Ausführung zufällig und unabhängig von der Wahl bisheriger Zufallsbits neu gewählt.
Die von probabilistischen Algorithmen mit polynomieller worst case Laufzeit und begrenzter Fehlerwahrscheinlichkeit lösbaren Probleme fasst man in der Komplexitätsklasse $\mathcal{BPP}$ (Bounded-Probability Polynomial Time) zusammen. Einen Algorithmus, dessen worst case Laufzeit polynomiell ist und der (mit einer begrenzten Wahrscheinlichkeit) falsche Ergebnisse liefern kann, nennt man auch *Monte-Carlo-Algorithmus*.

Definition 6 (Komplexitätsklasse $\mathcal{BPP}$) *Die Komplexitätsklasse $\mathcal{BPP}$ ist die Menge aller Entscheidungsprobleme $P \subseteq \{0,1\}^* \times \{0,1\}$, für die es eine probabilistische Turingmaschine M mit polynomieller worst case Laufzeit gibt, so dass für alle $x \in \{0,1\}^*$*

$$\Pr\left[(x, M(x)) \in P\right] \geq \frac{2}{3}$$

Der Ausdruck *Bounded-Probability* sagt aus, dass der Algorithmus eine Erfolgswahrscheinlichkeit größer als $\frac{1}{2}$ hat. Setzt man für die Erfolgswahrscheinlichkeit anstatt $\frac{2}{3}$ einen anderen Wert größer als $\frac{1}{2}$ ein, so ändert sich die definierte Klasse nicht. Die Klasse $\mathcal{BPP}$ besteht genau aus den Sprachen, die von einer probabilistischen Turing-Maschine mit polynomieller worst case Laufzeit und mit einer vernachlässigbar kleinen Fehlerwahrscheinlichkeit erkannt werden können. Wir benutzen den Begriff *vernachlässigbar* für Funktionen, die schneller verschwinden als der Kehrwert jedes Polynoms.

Definition 7 (Vernachlässigbare Funktion) *Eine Funktion $f : \mathbb{N} \to \mathbb{R}_0^+$ heißt vernachlässigbar, wenn es für jedes Polynom $p \in \mathbb{R}[x]$ ein $N \in \mathbb{N}$ gibt, so dass für alle $n \geq N$ gilt*

$$f(n) < \frac{1}{|p(n)|}$$

Beispiele für vernachlässigbare Funktionen sind $f(n) = 2^{-\sqrt{n}}$ und $f(n) = n^{-\log_2 n}$. Das Produkt aus einer vernachlässigbaren Funktion und einem Polynom ist wieder vernachlässigbar.

Betrachtet man Algorithmen mit einseitiger Fehlerwahrscheinlichkeit, so unterscheidet man ferner die Komplexitätsklassen $\mathcal{RP}$ und $co - \mathcal{RP}$ (siehe [57] für Details).
Das Gegenstück zu den Monte-Carlo-Algorithmen sind die *Las-Vegas-Algorithmen*. Bei den Las-Vegas-Algorithmen handelt es sich ebenfalls um randomisierte Algorithmen. Wir fordern aber, dass die Ausgaben eines solchen Algorithmus immer korrekt sind. Die Laufzeit eines Las-Vegas-Algorithmus ist eine Zufallsvariable, die von den Zufallsbits des Algorithmus abhängt. Man fordert, dass der Erwartungswert der Laufzeit polynomiell beschränkt ist und fasst alle Entscheidungsprobleme, für die es einen Las-Vegas-Algorithmus gibt, in der Komplexitätsklasse $\mathcal{EP}$ (Expected Polynomial Time) zusammen.

Bemerkung 1 *Als effizient lösbar oder leicht bezeichnen wir alle Probleme, für die es einen Algorithmus mit (erwartet) polynomieller Laufzeit und beschränkter Fehlerwahrscheinlichkeit gibt. Probleme, für die es keinen solchen Algorithmus gibt, bezeichnen wir als schwer. Den Begriff probabilistischer Algorithmus werden wir im folgenden als Oberbegriff für alle in diesem Abschnitt vorgestellten Arten von randomisierten Algorithmen verwenden.*

Ähnlich wie im deterministischen Fall kann man auch für probabilistische Algorithmen Reduktionen definieren.

Definition 8 (randomisiert polynomiell reduzierbar) *Ein Problem P_1 heißt randomisiert polynomiell reduzierbar auf ein Problem P_2, geschrieben $P_1 \leq_P^R P_2$, wenn es einen Algorithmus $\mathcal{A}$ mit erwartet polynomieller Laufzeit gibt, der P_1 löst. Algorithmus $\mathcal{A}$ kann dabei auf einen Algorithmus $\bar{\mathcal{A}}$, der P_2 löst, als Unterroutine zugreifen.*

Es gilt $\mathcal{P} \subseteq \mathcal{BPP}$. Es ist derzeit noch eine offene Frage, ob $\mathcal{P}$ eine echte Teilmenge von $\mathcal{BPP}$ ist. Ebenfalls offen ist, ob $\mathcal{BPP} \subseteq \mathcal{NP}$ gilt.

2.2.3 Anwendungen der Komplexitätstheorie in der Kryptografie

In der Kryptografie ist die $\mathcal{NP}$-Vollständigkeit eines Problems nur von begrenzter Aussagekraft, da sie sich auf die worst case Komplexität des Problems bezieht. So könnte es zum Beispiel sein, dass ein $\mathcal{NP}$-vollständiges Problem für wenige Instanzen schwer, für die Mehrheit der Instanzen jedoch leicht zu lösen ist. Geeigneter wären Aussagen über die average-case Komplexität von Problemen, die aussagt, wie schwer ein Problem für eine zufällig gewählte Instanz ist.

Ein Beispiel dafür, dass die Benutzung von $\mathcal{NP}$-vollständigen Problemen nicht unbedingt zu einem sicheren Verschlüsselungsverfahren führen muss, ist das Merkle-Hellman Knapsack Verfahren [36]. Dieses Verfahren beruht auf dem $\mathcal{NP}$-vollständigen Subset Sum Problem. Die zugrundeliegende Idee ist, eine leichte Instanz auszuwählen, die als geheimer Schlüssel dient, und diese dann in eine schwere Instanz zu transformieren, die den öffentlichen Schlüssel darstellt. Der erste Angriff mit polynomieller Laufzeit auf das Merkle-Hellman Knapsack Verfahren wurde im Jahre 1982 von Shamir [48] vorgestellt. Eine

Übersicht über die Entwicklung der Knapsack Verfahren gibt Odlyzko in [40].

Eine der grundlegenden Konstruktionen der Public-Key Kryptografie sind sogenannte *Einwegfunktionen*. Einwegfunktionen sind Funktionen, die effizient berechenbar aber hart zu invertieren sind.

Definition 9 (Einwegfunktion) *Eine Funktion $f : \{0,1\}^* \to \{0,1\}^*$ heißt Einwegfunktion, wenn folgende Bedingungen erfüllt sind:*

- *Einfach zu berechnen: Es gibt einen (deterministischen) Algorithmus $\mathcal{A}$ mit polynomieller Laufzeit, der auf Eingabe von $x \in \{0,1\}^*$ den Funktionswert $f(x)$ ausgibt (d.h. $\mathcal{A}(x) = f(x)$).*
- *Schwer zu invertieren: Für jeden probabilistischen Algorithmus $\mathcal{I}$ mit erwartet polynomieller Laufzeit ist die Inversionswahrscheinlichkeit*

$$\mathrm{P}(\mathcal{I}(1^n, f(x)) \in f^{-1}(f(x)))$$

  vernachlässigbar in n. Dabei wird die Inversionswahrscheinlichkeit über die Wahl des $x \in_R \{0,1\}^n$ und die Zufallbits von Algorithmus $\mathcal{I}$ gebildet.

Gilt zusätzlich $f(\{0,1\}^n) = \{0,1\}^n$ für alle $n \in \mathbb{N}$, so nennt man f Einwegpermutation.

Die Tatsache, dass f eine Einwegfunktion ist, sagt lediglich aus, dass für einen gegebenen Wert y aus dem Bildbereich von f Urbilder von y unter f nicht effizient berechnet werden können. Sie sagt aber nichts darüber aus, wie schwer es ist, gewisse Eigenschaften oder Teile des Urbildes zu berechnen (z.B. die unterwertigsten Bits). Dieser Umstand schränkt die Einsetzbarkeit von Einwegfunktionen für bestimmte kryptografische Anwendungen ein. Unter der Annahme, dass Einwegfunktionen existieren, lassen sich weitere Einwegfunktionen definieren, für die mit Kenntnis eines Bildes y bestimmte Eigenschaften des Urbildes von y unter f nicht effizient berechnet werden können (mit Kenntnis des Urbildes selbst aber effizient berechenbar sind). Diese Eigenschaften des Urbildes bezeichnet man als Hard-Core Prädikate der Funktion f.

Definition 10 (Hard-Core Prädikat) *Ein in polynomieller Laufzeit berechenbares Prädikat $b : \{0,1\}^* \to \{0,1\}$ heißt Hard-Core Prädikat der Funktion $f : \{0,1\}^* \to \{0,1\}^*$, wenn für jeden probabilistischen Algorithmus $\mathcal{A}$ mit erwartet polynomieller Laufzeit, jedes Polynom $p \in \mathbb{R}[x]$ und alle hinreichend großen $n \in \mathbb{N}$*

$$\mathrm{P}(\mathcal{A}(f(x)) = b(x)) < \frac{1}{2} + \frac{1}{p(n)}$$

gilt. Dabei wird die Wahrscheinlichkeit über die Wahl des $x \in_R \{0,1\}^n$ und die Zufallsbits von Algorithmus $\mathcal{A}$ gebildet.

Von einer Reihe von kryptografischen Primitiven, wie zum Beispiel von Pseudozufallsgeneratoren oder Pseudozufallsfunktionen, lässt sich zeigen, dass sie genau dann existieren, wenn Einwegfunktionen existieren (siehe [17]). Für Einwegfunktionen gibt es eine Reihe von Kandidaten, z.B. die diskrete Exponentiation in $\mathbb{Z}_p^*$ (siehe auch Abschnitt 2.3). Bisher konnte aber von keiner Funktion wirklich bewiesen werden, dass sie eine Einwegfunktion ist. Das Problem, für eine effizient berechenbare Funktion Urbilder zu berechnen, liegt in $\mathcal{NP}$. Wäre $\mathcal{P} = \mathcal{NP}$, so wäre auch die Inversion immer effizient möglich. Einwegfunktionen können also nur dann existieren, wenn $\mathcal{P} \neq \mathcal{NP}$. Die offene Frage, ob die Vermutung $\mathcal{P} \neq \mathcal{NP}$ korrekt ist, ist daher von wesentlicher Bedeutung für die Kryptografie.

2.2.4 Laufzeit von Algorithmen

Seien $n, u, v \in \mathbb{R}$ mit $n > e$. Dann schreibt man

$$L_n[u, v] = e^{v(\ln n)^u (\ln(\ln n))^{1-u}}$$

und benutzt diese Funktion zur Beschreibung der Laufzeit von Algorithmen zur Faktorisierung ganzer Zahlen und der Berechnung diskreter Logarithmen. Dabei entspricht n der zu faktorisierenden Zahl bzw. dem benutzten Modul p im Fall von diskreten Logarithmen (siehe auch Abschnitt 2.3). Die Eingabelänge des Algorithmus ist die binäre Länge $\lfloor \log_2 n \rfloor + 1 = \mathcal{O}(\log n)$ von n.

Für $u = 0$ erhält man

$$L_n[0, v] = e^{v(\ln n)^0 (\ln(\ln n))^1} = (\ln n)^v$$

Der betrachtete Algorithmus hat in diesem Fall polynomielle Laufzeit, wobei der Wert v den Grad des Polynoms angibt. Für $u = 1$ ergibt sich durch

$$L_n[0, v] = e^{v(\ln n)^1 (\ln(\ln n))^0} = e^{v \ln n} = n^v$$

eine exponentielle Laufzeit. Für $0 < u < 1$ hat der Algorithmus eine Laufzeit, die schneller als exponentiell aber langsamer als polynomiell ist. In diesem Fall spricht man von einer subexponentiellen Laufzeit.

2.3 Probleme und Annahmen in der Kryptografie

In diesem Abschnitt werden wir das Problem der Berechnung diskreter Logarithmen und die Berechnungs- und Entscheidungsvarianten des Diffie-Hellman Problems vorstellen. Die Annahme, dass diese Probleme in speziellen Gruppen schwer sind, gehört zu den ältesten und am besten untersuchten Sicherheitsannahmen in der Kryptografie.

Definition 11 (Diskreter Logarithmus) *Sei G eine endliche, zyklische Gruppe der Ordnung n, g ein Generator von G und $h \in G$. Der diskrete Logarithmus von h zur Basis g, geschrieben $\log_g h$, ist die eindeutig bestimmte ganze Zahl x, $0 \leq x \leq n-1$, so dass $h = g^x$ gilt.*

Definition 12 (Diskreter Logarithmus Problem (DLP)) *Sei eine zyklische Gruppe G der Ordnung n, ein Generator g von G und ein Element $h \in G$ gegeben. Finde die ganze Zahl x, $0 \leq x \leq n-1$, so dass $h = g^x$ gilt.*

Man nimmt an, dass das DLP nicht effizient lösbar ist in der multiplikativen Gruppe $\mathbb{Z}_p^*$, wobei p eine zufällig gewählte und hinreichend große Primzahl ist. Die Sicherheit zahlreicher kryptografischer Verfahren lässt sich beweisen, wenn man zugrunde legt, dass die folgende Annahme erfüllt ist.

Definition 13 (Diskreter Logarithmus Annahme) *Für eine zufällig gewählte Primzahl p mit Bitlänge k, einen zufällig gewählten Generator g von $\mathbb{Z}_p^*$ und ein zufällig gewähltes Gruppenelement $h \in_R \mathbb{Z}_p^*$ gibt es keinen Algorithmus mit in k polynomieller Laufzeit, der den diskreten Logarithmus $\log_g h$ mit nicht-vernachlässigbarer Wahrscheinlichkeit berechnen kann.*

Ähnliche Annahmen können auch für andere zyklische Gruppen formuliert werden. Auf Schnorr [47] geht die Idee zurück, zyklische Untergruppen primer Ordnung zu benutzen. Seien p, q Primzahlen und q ein Teiler von $p-1$. In diesem Fall hat $\mathbb{Z}_p^*$ genau eine Untergruppe G_q der Ordnung q. Jedes Element $g \in G_q$ ungleich der Eins erzeugt G_q und kann als Basis für den diskreten Logarithmus verwendet werden. Der Vorteil dieses Ansatzes ist, dass die verwendeten Exponenten wesentlich kleiner sind, was sich besonders bei modularen Exponentiationen in der Laufzeit bemerkbar macht. Auf der Schwierigkeit des DLP in solchen Untergruppen beruhen zum Beispiel die Identifikations- und Signaturschemata von Schnorr [47] und der DSA Signaturstandard [24]. Koblitz [22] und Miller [37] schlugen vor, zyklische Untergruppen primer Ordnung der (additiven) Gruppe der Punkte auf einer elliptischen Kurve zu benutzen.

Die effizientesten Algorithmen zur Berechnung von diskreten Logarithmen in beliebigen zyklischen Gruppen sind Pollard's Rho Methode [44] und der Baby-Step-Giant-Step Algorithmus [50] (auch bekannt als Shank's Algorithmus), die beide eine Laufzeit von $\mathcal{O}(\sqrt{q})$ Gruppenoperationen haben, wobei q die Ordnung der betrachteten zyklischen Gruppe ist. Während beim Baby-Step-Giant-Step Algorithmus $\mathcal{O}(\sqrt{q})$ Gruppenelemente gespeichert

werden müssen, erfordert Pollard's Rho Methode nur die Zwischenspeicherung weniger Werte. Algorithmen, die nur Gruppenoperationen und Tests auf Gleichheit von Gruppenelementen verwenden und deshalb auf beliebigen zyklischen Gruppen ausgeführt werden können, nennt man auch *generische Algorithmen*.
Der derzeit effizienteste Algorithmus in $\mathbb{Z}_p^*$ ist eine Variante des Zahlkörpersiebs (siehe [31]), die mit $L_n[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}}]$ eine subexponentielle Laufzeit hat. Die bekannten Algorithmen mit subexponentieller Laufzeit kann man nicht verwenden, um diskrete Logarithmen in Untergruppen G_q primer Ordnung von $\mathbb{Z}_p^*$ zu berechnen. In diesen Gruppen gibt es zwei mögliche Ansatzpunkte:

1. Die Berechnung diskreter Logarithmen in der Gruppe G_q mittels generischer Algorithmen. Diese haben eine Laufzeit von $\mathcal{O}(\sqrt{q})$ Gruppenoperationen.

2. Die Berechnung diskreter Logarithmen in $\mathbb{Z}_p^*$. Seien ein Generator g_1 von G_q, ein Generator g_2 von $\mathbb{Z}_p^*$ und $h \in G_q$ gegeben. Berechne die beiden Werte $y_1 = \log_{g_2} h$ und $y_2 = \log_{g_2} g_1$. Der gesuchte diskrete Logarithmus ergibt sich dann als $\log_{g_1} h = y_1 \cdot (y_2)^{-1}$.

Derzeit gelten Primzahlen der Länge 1024 Bits für $\mathbb{Z}_p^*$ und prime Untergruppen der Größe 160 Bits als untere Grenze für die Wahl der Parameter bei kryptografischen Verfahren, die auf der Schwierigkeit des DLP beruhen.
Eine ausführliche Übersicht über die zur Zeit effizientesten Algorithmen für diverse kryptografische Probleme und die daraus resultierenden empfohlenen Schlüssellängen findet sich in [32]. Gute Übersichtsartikel über das DLP sind [34] und [39].

Die folgende Bemerkung sagt aus, dass jeder Algorithmus, der diskrete Logarithmen zur Basis g_1 berechnen kann, auch zur Berechnung von diskreten Logarithmen zu einer beliebigen anderen Basis g_2 verwendet werden kann.

Bemerkung 2 (Die Schwierigkeit des DLP ist unabhängig vom Generator) *Seien g_1 und g_2 Generatoren einer zyklischen Gruppe G der Ordnung n und sei $h \in G$. Sei ferner $x = \log_{g_1} h$, $y = \log_{g_2} h$ und $z = \log_{g_1} g_2$. Dann gilt $g_1^x = h = g_2^y = g_1^{zy}$ und es folgt, dass $x = z \cdot y \pmod{n}$ gilt. Der gesuchte Wert y ergibt sich also als $y = x \cdot z^{-1} \pmod{n}$.*[3]

Eng verwandt mit dem Diskreter Logarithmus Problem ist das Diffie-Hellman Problem. Wir werden hier die Berechnungs- und die Entscheidungsvariante des Diffie-Hellman Problems vorstellen. Die Diffie-Hellman Probleme werden sich als höchstens so schwer wie das Problem der Berechnung diskreter Logarithmen herausstellen.

Definition 14 (Computational Diffie-Hellman Problem (CDHP)) *Seien eine zyklische Gruppe G der Ordnung n, ein Generator g von G und zwei Gruppenelemente $g^a, g^b \in G$, $1 \leq a, b < n$, gegeben. Finde g^{ab}.*

[3] Der Wert z ist invertierbar modulo n, andernfalls wäre g_2 kein Generator von G.

Das CDHP wurde von Diffie und Hellman im Jahre 1976 [13] eingeführt. In Gruppen, in denen das DLP effizient lösbar ist, kann auch das CDHP effizient gelöst werden. Eines der offenen Probleme der Kryptografie ist, ob die Umkehrung obiger Aussage gilt, d.h. ob das DLP und das CDHP äquivalent sind.
Die folgenden Aussagen, die auf den Boer [6] und Maurer [33] zurückgehen, zeigen die enge Verwandtschaft von DLP und CDHP:

1. Sei p eine Primzahl und die Primfaktorzerlegung von $p-1$ bekannt. Sei ferner $\phi(p-1)$ B-glatt für $B = \mathcal{O}((\ln p)^c)$ und eine Konstante c, d.h. $\phi(p-1)$ habe keine Primteiler größer als B. Dann sind das DLP und das CDHP in $\mathbb{Z}_p^*$ äquivalent.

2. Sei G eine endliche, zyklische Gruppe der Ordnung n und die Primfaktorzerlegung von n bekannt. Sei ferner $\phi(n)$ B-glatt für $B = \mathcal{O}((\ln n)^c)$ und eine Konstante c. Dann sind das DLP und das CDHP in G äquivalent.

3. Sei G eine endliche, zyklische Gruppe der Ordnung n und die Primfaktorzerlegung von n bekannt. Wenn für jeden Primteiler p von n, $p-1$ oder $p+1$ B-glatt sind, für $B = \mathcal{O}((\ln n)^c)$ und eine Konstante c, dann sind das DLP und das CDHP in G äquivalent.

Waldvogel und Massey [56] haben gezeigt, dass für zufällig und unabhängig gezogene Werte $a, b \in_R \{0, 1, \ldots, p-1\}$ der Wert $g^{ab} \pmod p$ annähernd uniform in $\mathbb{Z}_p^*$ verteilt sind.

Neben dem Diffie-Hellman Berechenbarkeitsproblem hat sich in der Kryptografie auch das zugehörige Entscheidungsproblem durchgesetzt.

Definition 15 (Decisional Diffie-Hellman Problem (DDHP)) *Seien eine zyklische Gruppe G der Ordnung n, ein Generator g von G und Gruppenelemente $g^a, g^b, g^c \in G$, $1 \leq a, b, c < n$, gegeben. Entscheide, ob $c = a \cdot b$ gilt.*

In Gruppen, in denen das CDHP effizient lösbar ist, kann auch das DDHP effizient gelöst werden. Zusammenfassend gilt also der folgende Satz.

Satz 2 *In zyklischen Gruppen G gilt: DDHP $\leq_P$ CDHP $\leq_P$ DLP.*

Um die Sicherheit kryptografischer Systeme charakterisieren zu können, definiert man wie im Fall des diskreten Logarithmus wieder Annahmen, die besagen, dass das CDHP und das DDHP für zufällig gewählte Instanzen schwer sind.

Definition 16 (Computational Diffie-Hellman Annahme) *Es gibt keinen effizienten Algorithmus, der auf Eingabe einer zyklischen Gruppe G der Ordnung n, eines zufällig gewählten Generators g von G und zweier zufällig gewählter Gruppenelemente $g^a, g^b \in G$ das Gruppenelement g^{ab} mit nicht-vernachlässigbarer Wahrscheinlichkeit berechnen kann.*

Definition 17 (Decisional Diffie-Hellman Annahme) *Es gibt keinen effizienten Algorithmus, der auf Eingabe einer zyklischen Gruppe G der Ordnung n, eines zufällig gewählten Generators g von G und zufällig gewählter Gruppenelemente $g^a, g^b, g^c \in G$ mit Wahrscheinlichkeit größer $\frac{1}{2} + v(n)$ mit einem nicht-vernachlässigbaren Vorteil $v(n)$ zwischen g^{ab} und g^c, für zufällig gewählte $a, b, c \in_R \{1, \ldots n-1\}$, unterscheiden kann.*

Shoup [51] hat gezeigt, dass das DDHP schwer ist für jeden generischen Algorithmus, d.h. für Algorithmen, die nur Gruppenoperationen ausführen können und keinen Zugriff auf die Darstellung der Gruppenelemente haben. Dass es aber in bestimmten in der Kryptografie benutzten Gruppen auch nicht-generische Algorithmen gibt, die das DDHP für zufällige Instanzen mit nicht-vernachlässigbarem Vorteil lösen, zeigt die folgende Bemerkung.

Bemerkung 3 *Obwohl man annimmt, dass die Diskreter Logarithmus Annahme in $\mathbb{Z}_p^*$ erfüllt ist, kann man zeigen, dass die Decisional Diffie-Hellman Annahme in dieser Gruppe nicht gilt.*
Sei g ein Generator von $\mathbb{Z}_p^*$. Dann ist g auf jeden Fall kein quadratischer Rest. Werden a, b, c zufällig gewählt, so sind g^a, g^b, g^c jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ quadratische Reste. Ist g^a oder g^b ein quadratischer Rest, so muss dies auch für g^{ab} gelten. Analog gilt, dass wenn g^a und g^b keine quadratischen Reste sind, auch g^{ab} kein quadratischer Rest sein kann. Da in $\mathbb{Z}_p^*$ effizient geprüft werden kann, ob Elemente quadratische Reste sind, kann durch diese Überlegungen bereits für die Hälfte aller Instanzen das DDHP entschieden werden.

Die Decisional Diffie-Hellman Annahme gilt hingegen als akzeptiert für Untergruppen primer Ordnung G_q von $\mathbb{Z}_p^*$. Dieses Vertrauen wird unter anderem dadurch gerechtfertigt, dass folgende worst-case Version der Decisional Diffie-Hellman Annahme sich für G_q auf Definition 17 reduzieren lässt, in der sie als average-case Annahme formuliert wurde (siehe auch [38]):

Es gibt keinen effizienten Algorithmus, der auf Eingabe (p, q, g, g^a, g^b, g^c) für jedes $g \in G_q$ und jede $a, b, c \in \mathbb{Z}_q$ mit Wahrscheinlichkeit $1 - v(n)$, wobei $v(n)$ vernachlässigbar ist, entscheiden kann, ob $c = a \cdot b$.

In G_q ist das Decisional Diffie-Hellman Problem im average-case also mindestens so schwer wie im worst-case. Es folgt, dass das DDHP in G_q entweder schwer im average-case oder leicht im worst-case ist. Da alle Bemühungen, das DDHP in G_q zu lösen, bisher erfolglos geblieben sind, stärkt obige Annahme das Vertrauen in die Decisional Diffie-Hellman Annahme für G_q.

Okamoto und Pointcheval [41] haben den Begriff der Gap-Gruppen eingeführt. Das sind Gruppen, in denen das Decisional Diffie-Hellman Problem leicht, das Computational Diffie-Hellman Problem jedoch schwer ist. Ihre Konstruktion benutzt die Tate-Pairing auf speziellen hyper-elliptischen Kurven. Sie zeigen ferner, wie man mittels Gap-Gruppen neue Signaturverfahren konstruieren kann.

2.4 Diffie-Hellman Schlüsselaustausch

Das Protokoll von Diffie und Hellman zur Schlüsselvereinbarung [13] erlaubt es zwei Parteien, sich durch den Austausch von Nachrichten über einen offenen Kanal auf ein gemeinsames Geheimnis zu einigen. Die Sicherheit des Verfahrens beruht auf der Schwierigkeit des Computational Diffie-Hellman Problems.

Zunächst einigen sich beide Parteien auf eine hinreichend große Primzahl p und einen Generator g von $\mathbb{Z}_p^*$. Das Protokoll besteht aus den folgenden zwei Schritten, die parallel ausgeführt werden können:

- Teilnehmer A wählt zufällig einen Wert a, $1 \leq a \leq p-2$ und sendet $g^a \bmod p$ an Teilnehmer B.
- Teilnehmer B wählt zufällig einen Wert b, $1 \leq b \leq p-2$ und sendet $g^b \bmod p$ an Teilnehmer A.

Nach Erhalt der Nachricht kann jeder der Teilnehmer das gemeinsame Geheimnis $k \in \mathbb{Z}_p^*$ durch Exponentiation des erhaltenen Wertes mit der von ihm gewählten Zufallszahl berechnen: $k = (g^a)^b \bmod p = (g^b)^a \bmod p$.

Ein passiver Angreifer kann genau dann den gemeinsamen Schlüssel k aus den abgehörten Werten $g^a \bmod p$ und $g^b \bmod p$ berechnen, wenn er das Computational Diffie-Hellman Problem in $\mathbb{Z}_p^*$ lösen kann.
Das Diffie-Hellman Protokoll ist aber nicht sicher gegen aktive Angreifer, d.h. gegen Angreifer die die ausgetauschten Nachrichten sowohl mitlesen als auch verändern können. Bei Benutzung dieses Protokolls sollte deshalb ein zusätzliches Verfahren eingesetzt werden, dass die Authentizität der versandten Nachrichten sichert.

Diffie-Hellman kann außer in $\mathbb{Z}_p^*$ auch in jeder anderen zyklischen Gruppe ausgeführt werden, in der Gruppenoperationen effizient ausgeführt werden können und in der das Computational Diffie-Hellman Problem schwer ist.
Die Teilnehmer können die Werte $g^a \bmod p$ bzw. $g^b \bmod p$ auch als öffentlichen Schlüssel festlegen und mittels signierter Zertifikate verteilen. Das Geheimnis k ist dann für die Gültigkeitsdauer der öffentlichen Schlüssel fest.
Ist der gewählte Exponent a (bzw. b) nicht teilerfremd zur Gruppenordnung $p-1$ von $\mathbb{Z}_p^*$, so ist $g^a \bmod p$ (bzw. $g^b \bmod p$) kein Generator von $\mathbb{Z}_p^*$. In dem Fall wird die Menge der Werte, die k annehmen kann, unnötig eingeschränkt. Deshalb ist es ratsam, Werte $a, b \in_R \mathbb{Z}_{p-1}^*$ zu wählen. Um Angriffe zu vermeiden, sollte jeder Teilnehmer prüfen, ob der erhaltene Wert wirklich ein Generator von $\mathbb{Z}_p^*$ ist.

2.5 ElGamal

Das erstmals im Jahre 1984 von ElGamal vorgestellte asymmetrische Verschlüsselungsverfahren [15] beruht auf der Schwierigkeit des Computational Diffie-Hellman Problems.

Bei der Schlüsselerzeugung wählt sich jeder Teilnehmer eine hinreichend große Primzahl p, einen zufälligen Generator g von $\mathbb{Z}_p^*$ und eine Zufallszahl $x \in_R \{1, \ldots, p-2\}$. Die Zufallszahl x ist der geheime Schlüssel des Teilnehmers. Der öffentliche Schlüssel ist das Tupel (p, g, h), wobei $h = g^x \bmod p$ ist.

Um eine Nachricht $m \in \mathbb{Z}_p^*$ bzgl. des öffentlichen Schlüssels (p, g, h) zu verschlüsseln, macht man folgendes:

1. Wähle eine Zufallszahl $r \in_R \{1, \ldots, p-2\}$.
2. Berechne $C_1 = m \cdot h^r \bmod p$ und $C_2 = g^r \bmod p$.
3. Der Geheimtext von m ist $C(m) = (C_1, C_2)$.

Um den Geheimtext $C(m)$ zu entschlüsseln und den zugehörigen Klartext m zu erhalten, wird mittels des geheimen Schlüssels x der Wert h^r aus der zweiten Komponente des Geheimtextes C_2 berechnet. Die gesuchte Nachricht m ergibt sich als $m = C_1 \cdot (C_2)^{-x} \bmod p$.

Das ElGamal Verfahren ist eng verwandt mit dem Diffie-Hellman Schlüsselaustausch. Zunächst generiert man mit $h^r = g^{xr} \bmod p$ einen Sitzungsschlüssel, der anschließend mit der zu verschlüsselnden Nachricht multipliziert wird.

Durch die Wahl einer neuen Zufallszahl r für jede Verschlüsselung wird sichergestellt, dass zwei Verschlüsselungen der selben Nachricht nur mit exponentiell kleiner Wahrscheinlichkeit gleich sind.
Für die Sicherheit von ElGamal ist es notwendig, dass für jede Verschlüsselung eine neue Zufallszahl gewählt wird. Angenommen, die Zufallszahl r wird benutzt, um sowohl die Nachrichten m_1 und m_2 zu verschlüsseln. Seien (C_1, C_2) und $(\widehat{C}_1, C_2)$ die zugehörigen Geheimtexte. Ein Angreifer kann in diesem Fall $C_1 \cdot (\widehat{C}_1)^{-1} \equiv m_1 \cdot (m_2)^{-1} \pmod{p}$ berechnen. Insbesondere kann also bei Kenntnis einer der beiden Nachrichten die jeweils andere berechnet werden.

Das Problem, bei gegebenem öffentlichen Schlüssel p, g, h und einem ElGamal Geheimtext (C_1, C_2) die verschlüsselte Nachricht m vollständig zu berechnen, ist äquivalent zum Computational Diffie-Hellman Problem (siehe etwa [35]).

Neben $\mathbb{Z}_p^*$ kann man auch jede andere zyklische Gruppe für das ElGamal Verfahren benutzen, in der das Diskreter Logarithmus Problem schwer ist. Benutzt man Untergruppen G_q primer Ordnung von $\mathbb{Z}_p^*$, so ist ElGamal semantisch sicher unter der Decisional

Diffie-Hellman Annahme, d.h. es gibt keinen effizienten Algorithmus, der nach Wahl von $m_1, m_2 \in G_q$ mit einer Wahrscheinlichkeit, die nicht-vernachlässigbar größer als $\frac{1}{2}$ ist, Verschlüsselungen C_1 und C_2 von m_1 bzw. m_2 unterscheiden kann.

Die Einschränkung, dass die verschlüsselte Nachricht immer ein Element der benutzten zyklischen Gruppe sein muss, ist manchmal hinderlich. Wünschenswert wäre vielmehr ein Verfahren, bei dem beliebige Bitstrings verschlüsselt werden können.
Dies kann durch Benutzung einer Hashfunktion erreicht werden. Sei $m \in \{0,1\}^k$ die zu verschlüsselnde Nachricht und $H : G \to \{0,1\}^k$ eine pair-wise independent Hashfunktion[4], wobei G die benutzte zyklische Gruppe ist. Der Geheimtext von m hat dann die Form $C(m) = (C_1, C_2) = (m \oplus H(h^r), g^r)$. Mittels des geheimen Schlüssels x kann der Empfänger $m = C_1 \oplus H(C_2^x)$ berechnen (siehe auch [38]).

2.6 Innere Automorphismen

In diesem Abschnitt werden wir uns mit der Gruppe der inneren Automorphismen beschäftigen und zwei neue Berechnungsprobleme einführen, die später zur Konstruktion eines asymmetrischen Verschlüsselungsverfahrens benutzt werden. Um diese Probleme anschließend analysieren zu können, brauchen wir die Begriffe Zentrum und Zentralisator.

Definition 18 (Zentrum) *Sei G eine Gruppe. Das Zentrum $Z(G)$ von G ist definiert als*

$$\begin{aligned} Z(G) &= \{g \in G \mid hg = gh \ \ \forall h \in G\} \\ &= \{g \in G \mid ghg^{-1}h^{-1} = e_G \ \ \forall h \in G\}. \end{aligned}$$

Das Zentrum $Z(G)$ ist ein Normalteiler der Gruppe G. Das Zentrum misst den Grad der Kommutativität von G. Es ist $Z(G) = G$ genau dann, wenn G abelsch ist.

Definition 19 (Zentralisator) *Sei G eine Gruppe und $g \in G$. Der Zentralisator von g ist definiert als*

$$Z(g) = \{h \in G \mid gh = hg\}.$$

Der Zentralisator $Z(g)$ enthält also alle Gruppenelemente, die mit g kommutieren. Das Zentrum einer Gruppe ist der Durchschnitt aller Zentralisatoren, d.h. $Z(G) = \bigcap_{g \in G} Z(g)$.

In Abschnitt 5.5.4 verwenden wir die Begriffe *Zentrum* und *Zentralisator* auch für Ringe bzw. Ringelemente. Für einen Ring R und ein Ringelement $r \in R$ definieren wir $Z(R) := \{r \in R \mid sr = rs \ \forall s \in R\}$ und $Z(r) := \{s \in R \mid sr = rs\}$.
In einigen Fällen geht aus dem Kontext nicht klar hervor, auf welche Struktur Bezug genommen wird. Für $g \in GL(2,R) \subseteq M(2,R)$ kann sich der Zentralisator $Z(g)$ im Ring

[4]Bei einer pair-wise independent Hashfunktion $H : U \to V$ fordert man, dass für beliebige $u_1, u_2 \in V$ mit $u_1 \neq u_2$ und beliebige $v_1, v_2 \in V$ die Bedingung $\mathrm{P}(H(u_1) = v_1 \text{ und } H(u_2) = v_2) = \frac{1}{|V|^2}$ erfüllt ist. Ein anderer Name für pair-wise independent ist (strongly) universal.

$M(2,R)$ etwa vom Zentralisator $Z(g)$ in der multiplikativen Gruppe $GL(2,R)$ unterscheiden. In diesen Fällen wird die zugrundeliegende Struktur als zusätzlicher Index angegeben, z.B. $Z_{M(2,R)}(g) = \{h \in M(2,R) \mid gh = hg\}$ und $Z_{GL(2,R)}(g) = \{h \in GL(2,R) \mid gh = hg\}$.

Definition 20 (Innerer Automorphismus) *Sei G eine Gruppe. Dann ist mit*

$$\begin{aligned} Inn: \; G &\to Aut(G) \\ g &\mapsto Inn(g) \end{aligned}$$

eine homomorphe Abbildung von der Gruppe G in die Automorphismengruppe von G gegeben durch $Inn(g)(h) = ghg^{-1}$.

Wir nennen $Inn(g)$ einen inneren Automorphismus. Zusammen mit der Hintereinanderausführung bildet die Menge $Inn(G) = \{Inn(g) \mid g \in G\}$ aller inneren Automorphismen einer Gruppe G wieder eine Gruppe, die Gruppe der inneren Automorphismen.

Ist G eine abelsche Gruppe, so ist $Inn(g)$ die Identität auf G für alle $g \in G$ und $Inn(G)$ ist trivial. Man sieht direkt, dass $ker(Inn) = Z(G)$ gilt.

Definition 21 (Erzeugendensystem) *Sei G eine Gruppe und $S \subseteq G$. Die Gruppe $<S> := \bigcap \{U \mid U$ ist Untergruppe von G mit $S \subseteq U\}$ heißt die von S erzeugte Untergruppe von G. Ist $G = <S>$, dann heißt G von S erzeugt und S ein Erzeugendensystem von G. G heißt endlich erzeugt, wenn es ein endliches Erzeugendensystem $\{\gamma_1, \ldots, \gamma_n\} \subseteq G$ gibt. In diesem Fall schreibt man $G = <\gamma_1, \ldots, \gamma_n>$.*

Die Tatsache, dass sich in einer endlich erzeugten Gruppe jedes Element als Produkt der Generatoren γ_i, $1 \leq i \leq n$, schreiben lässt, nutzen wir aus, um innere Automorphismen $Inn(g)$ zu definieren, ohne das konjugierende Element g offenlegen zu müssen. Aufgrund der Homomorphie-Eigenschaft ist $Inn(g)$ bereits eindeutig bestimmt, wenn die Bilder $Inn(g)(\gamma_i)$ der Generatoren von G unter $Inn(g)$ gegeben sind.

2.6.1 Die Kojugationsprobleme

In diesem Abschnitt werden mit dem Konjugationsproblem und dem speziellen Konjugationsproblem zwei Probleme definieren, die uns die Konstruktion kryptografischer Protokolle auf nicht-abelschen Gruppen erlauben.

Definition 22 (Konjugationsproblem (CP)) *Sei G eine Gruppe und seien $x, y \in G$ mit $y = gxg^{-1}$ für ein $g \in G$. Finde ein $\bar{g} \in G$, so dass $\bar{g}x\bar{g}^{-1} = y$ gilt.*

Das Konjugationsproblem hat im allgemeinen keine eindeutige Lösung. Sei mit $x, y \in G$ eine Instanz des Konjugationsproblems gegeben. Sei ferner $g \in G$ eine Lösung dieser Instanz, d.h. $y = gxg^{-1}$. Dann lösen alle Elemente der Menge $g \cdot Z(x)$ ebenfalls diese Instanz. Der folgende Satz zeigt, dass es keine weiteren Lösungen gibt.

Satz 3 *Sei $x, y \in G$ eine Instanz des Konjugationsproblems in G und $g \in G$ eine Lösung dieser Instanz, d.h. es gilt $y = gxg^{-1}$. Dann ist die Menge L aller Lösungen dieser Instanz gegeben durch $L = g \cdot Z(x) = Z(y) \cdot g$.*

Beweis: Wir zeigen zunächst, dass $L = g \cdot Z(x)$ gilt. Sei $h \in g \cdot Z(x)$, d.h. es gibt ein $z \in Z(x)$ mit $h = gz$. Dann gilt $hxh^{-1} = gzxz^{-1}g^{-1} = gx(zz^{-1})g^{-1} = gxg^{-1} = y$. Es folgt, dass $g \cdot Z(x) \subseteq L$ erfüllt ist. Wir nehmen nun an, dass $h \in L$ ist, d.h.

$$\begin{aligned} & y = gxg^{-1} = hxh^{-1} \\ \Leftrightarrow \quad & g^{-1}hx = xg^{-1}h \\ \Leftrightarrow \quad & g^{-1}h \in Z(x) \\ \Leftrightarrow \quad & h \in g \cdot Z(x) \end{aligned}$$

Also gilt auch $L \subseteq g \cdot Z(x)$ und es folgt $L = g \cdot Z(x)$. Der Beweis der Behauptung $L = Z(y) \cdot g$ verläuft analog. □

In Abschnitt 2.6 wurde gezeigt, dass innere Automorphismen in endlich erzeugten Gruppen durch die Bilder der Generatoren von G bereits eindeutig definiert sind. Neben der Angabe des definierenden Elements g eines inneren Automorphismus $Inn(g)$ gibt es also weitere Arten der Darstellung für $Inn(g)$. Das spezielle Konjugationsproblem besteht darin, zu einem gegebenen inneren Automorphismus ein definierendes Element zu finden.

Definition 23 (Spezielles Konjugationsproblem (SCP)) *Seien eine Gruppe G und ein innerer Automorphismus $\varphi \in Inn(g)$ gegeben. Finde ein $g \in G$ mit $Inn(g) = \varphi$.*

Genau wie das Konjugationsproblem hat auch das spezielle Konjugationsproblem keine eindeutige Lösung.

Satz 4 *Es gilt $Inn(g) = Inn(h) \Leftrightarrow g = h \cdot z$ mit $z \in Z(G)$.*

Beweis: Es seien $g, h \in G$ mit $Inn(g) = Inn(h)$. Mit der Aussage von Satz 3 folgt wegen $Z(G) = \cap_{g \in G} Z(g)$, dass g dann von der Form $g = h \cdot z$ mit $z \in Z(G)$ sein muss.
Sei nun $g = h \cdot z$ mit $z \in Z(G)$. Dann gilt $g \cdot x \cdot g^{-1} = h \cdot z \cdot x \cdot z^{-1} \cdot h^{-1} = h \cdot x \cdot h^{-1}$ für alle $x \in G$ und damit $Inn(g) = Inn(h)$. □

Die Schwierigkeit des speziellen Konjugationsproblems hängt stark davon ab, wie die Abbildung $Inn(g)$ gegeben ist. Ist $Inn(g)$ z.B. durch das definierende Element g gegeben, so ist das SCP trivial. Wir werden bei der Kryptanalyse des MOR Systems in den folgenden Kapiteln Algorithmen vorstellen, die nicht spezielle Darstellungen der Abbildung $Inn(g)$ ausnutzen. In diesem Fall kann man sich $Inn(g)$ als Black-Box vorstellen. Ein Angreifer kann beliebige Gruppenelemente h in die Black-Box füttern und erhält den Funktionswert $Inn(g)(h)$ als Antwort. Die vorgestellten Angriffe funktionieren also auch noch, wenn eine andere Darstellung für die gegebenen inneren Automorphismen gewählt wird.

Bemerkung 4 *In vielen Gruppen lässt sich das spezielle Konjugationsproblem auf das Konjugationsproblem reduzieren. Sei G eine endlich erzeugte Gruppe mit den Generatoren $\gamma_1, \ldots, \gamma_n$. Sei ferner eine Abbildung $Inn(g)$ und ein Algorithmus $\mathcal{A}$, der das Konjugationsproblem in G löst, gegeben. Berechne die Bilder $Inn(g)(\gamma_i)$ aller Generatoren und gib die Instanzen $\gamma_i, Inn(g)(\gamma_i)$ des Konjugationsproblem in G an $\mathcal{A}$. Seien $g_1, \ldots, g_n$ die von $\mathcal{A}$ ausgegebenen Lösungen dieser Instanzen. Die Lösungsmenge L der gegebenen Instanz des speziellen Konjugationsproblems ist dann $L = \bigcap_{i=1}^{n} g_i \cdot Z(\gamma_i)$.*

2.7 Lineare Gruppen

Ist K ein Körper, so bezeichnet man mit $M(m \times n, K)$ die Menge aller $m \times n$ Matrizen mit Einträgen aus K. Die Menge $M(m \times n, K)$ ist zusammen mit der komponentenweise Addition und der skalaren Multiplikation ein Vektorraum über K. Für quadratische Matrizen schreibt man auch $M(n, K)$ statt $M(n \times n, K)$.

Im Folgenden werden vor allem die allgemeine und spezielle lineare Gruppe für uns von Interesse sein. Die allgemeine lineare Gruppe $GL(n, K) = \{A \in M(n \times n, K) \mid \det(A) \neq 0\}$ ist die Menge aller invertierbaren $n \times n$ Matrizen über K. Die Menge $GL(n, K)$ bildet zusammen mit der gewöhnlichen Matrixmultiplikation eine Gruppe. Die spezielle lineare Gruppe $SL(n, K) = \{A \in M(n \times n, K) \mid \det(A) = 1\}$ ist eine Untergruppe von $GL(2, K)$.

2.7.1 Die linearen Gruppen $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$

In diesem Abschnitt fassen wir Eigenschaften der Gruppen $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$ zusammen, die im Folgenden noch benötigt werden. Mit δ_{ij} bezeichnen wir Matrizen, deren (i, j) - Eintrag den Wert 1 und deren übrige Komponenten den Wert 0 haben.

Die beiden Matrizen

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

erzeugen die Gruppe $SL(2, \mathbb{Z})$ und damit auch $SL(2, \mathbb{Z}_p)$ (siehe [59]), d.h. jede Matrix $M \in SL(2, \mathbb{Z}_p)$ lässt sich schreiben als

$$M = S^{i_0} T^{j_1} S T^{j_2} \ldots S T^{j_n} S^{i_{n+1}}$$

mit $n \in \mathbb{N}$, $i_0, i_{n+1} \in \{0, 1\}$ und $j_k \in \mathbb{Z}_p$ für $1 \leq k \leq n$.

Matrizen $M \in SL(2, \mathbb{Z}_p)$, deren $(2, 1)-$ Eintrag ungleich Null ist, lassen sich darstellen als

$M = T^{j_1}ST^{j_2}ST^{j_3}$ mit $j_1, j_2, j_3 \in \mathbb{Z}_p$. Diese Darstellung kann effizient berechnet werden:

$$\begin{aligned} T^{j_1}ST^{j_2}ST^{j_3} &= \begin{pmatrix} 1 & j_1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & j_2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & j_3 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} j_1j_2 - 1 & j_1j_2j_3 - j_3 - j_1 \\ j_2 & j_2j_3 - 1 \end{pmatrix} \end{aligned}$$

Für $M = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix}$ mit $m_3 \neq 0$ ergibt sich also $j_2 = m_3$, $j_1 = (m_1 + 1)j_2^{-1}$ und $j_3 = (m_4+1)j_2^{-1}$. Um eine Darstellung von M bzgl. der Generatoren S und T zu berechnen, sind also lediglich 1 Inversion und 2 Multiplikationen notwendig.

Satz 5 *Es gilt:*

i) $|SL(2, \mathbb{Z}_p)| = (p-1)(p+1)p$,

ii) $|GL(2, \mathbb{Z}_p)| = (p-1)^2(p+1)p$,

iii) $Z(SL(2, \mathbb{Z}_p)) = \pm I$,

iv) $Z(GL(2, \mathbb{Z}_p)) = \{c \cdot I \mid c \in \mathbb{Z}_p^*\}$.

Beweis: Siehe [25].

2.7.2 Das DLP in Untergruppen von $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$

In diesem Abschnitt soll das Diskreter Logarithmus Problem in den Gruppen $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$ betrachtet werden. In Abschnitt 2.3 wurde das Diskreter Logarithmus Problem für zyklische Gruppen definiert. Da weder $SL(2, \mathbb{Z}_p)$ noch $GL(2, \mathbb{Z}_p)$ zyklisch sind, ist diese Definition zunächst nur auf zyklische Untergruppen anwendbar. Unter dem Diskreter Logarithmus Problem in $SL(2, \mathbb{Z}_p)$ (bzw. $GL(2, \mathbb{Z}_p)$) sei im folgenden das Problem gemeint, auf Eingabe M_1, M_2 mit $M_1 \in SL(2, \mathbb{Z}_p)$ (bzw. $M_1 \in GL(2, \mathbb{Z}_p)$) und $M_2 \in <M_1>$ die ganze Zahl x, $0 \leq x \leq ord(M_1) - 1$, zu finden, so dass $(M_1)^x = M_2$ gilt.

Satz 6 *Es gilt* $DLP_{\mathbb{Z}_p^*} \leq_P DLP_{SL(2,\mathbb{Z}_p)}$.

Beweis: Sei $g, h = g^x \bmod p$ eine Instanz des DLP in $\mathbb{Z}_p^*$ und $\mathcal{A}$ ein Algorithmus, der das DLP in $SL(2, \mathbb{Z}_p)$ löst. Der folgende Algorithmus löst dann das DLP in $\mathbb{Z}_p^*$:

- Berechne $\bar{g} = g^{-1} \pmod p$ und $\bar{h} = h^{-1} \pmod p$.
- Setze $M_1 := \begin{pmatrix} g & 0 \\ 0 & \bar{g} \end{pmatrix}$ und $M_2 := \begin{pmatrix} h & 0 \\ 0 & \bar{h} \end{pmatrix}$.
  Nach Definition von $\bar{g}$ und $\bar{h}$ gilt $M_1, M_2 \in SL(2, \mathbb{Z}_p)$ mit $\log_g h = \log_{M_1} M_2$.
- Gib das Paar (M_1, M_2) als Eingabe an $\mathcal{A}$ und mache die selbe Ausgabe wie $\mathcal{A}$. □

Korollar 1 *Es gilt* $DLP_{\mathbb{Z}_p^*} \leq_P DLP_{GL(2,\mathbb{Z}_p)}$.

Beweis: Da $SL(2,\mathbb{Z}_p) \subset GL(2,\mathbb{Z}_p)$ gilt $DLP_{SL(2,\mathbb{Z}_p)} \leq_P DLP_{GL(2,\mathbb{Z}_p)}$. Die Behauptung folgt aus den Sätzen 1 und 6. □

Wir beschäftigen uns nun mit der Frage, ob ein Angreifer, der diskrete Logarithmen in $\mathbb{Z}_p^*$ berechnen kann, damit auch bereits diskrete Logarithmen in $GL(2,\mathbb{Z}_p)$ berechnen kann.
Sei $g \in GL(2,\mathbb{Z})$ und $G = <g>$ die von g erzeugte zyklische Untergruppe von $GL(2,\mathbb{Z})$. Für $\det(g) \neq 1$ liefert der Determinantenmultiplikationssatz erste Anhaltspunkte über den gesuchten diskreten Logarithmus: Sind $g, g^k \in GL(2,\mathbb{Z}_p)$ gegeben, so ist $\det(g^k) \equiv (\det(g))^k \pmod{p}$. Kann ein Angreifer diskrete Logarithmen in der von $\det(g)$ erzeugten Untergruppe von $\mathbb{Z}_p^*$ berechnen, so erhält er bereits $k \pmod{t}$, wobei t die Ordnung von $\det(g)$ in $\mathbb{Z}_p^*$ bezeichnet. Die Matrix g sollte daher so gewählt werden, dass das Diskreter Logarithmus Problem schwer in der von $\det(g)$ erzeugten Untergruppe von $\mathbb{Z}_p^*$ ist. Man umgeht dieses Problem, wenn man die Wahl von g auf $SL(2,\mathbb{Z})$ einschränkt. In diesem Fall liefert die Determinante keinerlei Informationen mehr über den gesuchten diskreten Logarithmus.
Ähnliche Aussagen lassen sich über eine Betrachtung der Eigenwerte machen. Ist e ein Eigenwert der Matrix g, so ist e^k ein Eigenwert von g^k. Die Eigenwerte von $g \in GL(2,\mathbb{Z}_p)$ liegen in einem zu $GF(p^2)$ isomorphen Erweiterungskörper K von $\mathbb{Z}_p^*$. Können in K diskrete Logarithmen berechnet werden, so erhält man den gesuchten diskreten Logarithmus in $GL(2,\mathbb{Z}_p)$ modulo der Ordnung der Eigenwerte in K.

In Satz 6 haben wir gezeigt, dass das Diskreter Logarithmus Problem in $SL(2,\mathbb{Z}_p)$ im worst case mindestens so schwer ist, wie das Diskreter Logarithmus Problem in $\mathbb{Z}_p^*$. Eine unglückliche Wahl des Generators $g \in SL(2,\mathbb{Z}_p)$ kann jedoch trotzdem dazu führen, dass diskrete Logarithmen in $G = <g>$ effizient berechnet werden können.

In [42] wird eine Matrix der Form $g = A(I + c\delta_{12})A^{-1}$ mit $A \in SL(2,\mathbb{Z}_p)$ und $c \in \mathbb{Z}_p$, $c \neq 0$, als Generator vorgeschlagen, um eine Untergruppe der Ordnung p zu erhalten.
Sei $A = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \in SL(2,\mathbb{Z}_p)$. Betrachtet man die Potenzen von $g = A(I + c\delta_{12})A^{-1}$, so erhält man

$$\begin{aligned} g^n &= A(I + c\delta_{12})^n A^{-1} \\ &= A(I + cn\delta_{12})A^{-1} \\ &= \begin{pmatrix} 1 - nca_1a_3 & nc(a_1)^2 \\ -nc(a_3)^2 & 1 + nca_1a_3 \end{pmatrix} \end{aligned}$$

Als $(1,2)$-Komponenten von g bzw. g^n erhält man ca_1^2 und nca_1^2. Durch Division erhält man den gesuchten diskreten Logarithmus.

Der folgende Satz aus [3] sagt aus, dass das Diskreter Logarithmus Problem leicht ist in allen zyklischen Untergruppen von $SL(2, \mathbb{Z}_p)$ der Ordnung p.

Satz 7 *Sei $A \in SL(2, \mathbb{Z}_p)$ mit $ord(A) = p$, dann kann das Diskreter Logarithmus Problem in der von A erzeugten Gruppe $< A >$ effizient gelöst werden.*
Seien a_{ij} und $a_{ij}^{(k)}$ die (i,j)-Komponenten von A bzw. A^k. Der gesuchte diskrete Logarithmus ergibt sich dann als $k = a_{12}^{(k)} \cdot (a_{12})^{-1}$ bzw. als $k = a_{21}^{(k)} \cdot (a_{21})^{-1}$ für den Fall, dass $a_{12} = 0$ gilt. Da $A \in SL(2, \mathbb{Z}_p)$ mit $ord(A) = p$ nicht diagonalisierbar sind, kann der Fall $a_{12} = a_{21} = 0$ nicht eintreten.

Beweis: Siehe [3].

Um trotzdem Untergruppen primer Ordnung von $GL(2, \mathbb{Z}_p)$ und $SL(2, \mathbb{Z}_p)$ zu erhalten, in denen das DLP schwer ist, nutzen wir folgende Konstruktion:

Satz 8 *Seien p, q prim mit $q \mid p-1$ und sei G_q die eindeutig bestimmte Untergruppe von $\mathbb{Z}_p^*$ der Ordnung q. Sei ferner $\mathcal{M} := \left\{ \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \in GL(2, \mathbb{Z}_p) \mid a, d \in G_q, a \neq d, b \in \mathbb{Z}_p \right\}$. Dann gilt:*

i) Die Elemente von $\mathcal{M}$ haben alle Ordnung q.

ii) $DLP_{G_q} =_P DLP_{\mathcal{M}}$

Beweis: Sei $M \in \mathcal{M}$. Durch vollständige Induktion lässt sich zeigen, dass

$$M^n = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix}^n = \begin{pmatrix} a^n & b(\sum_{i=0}^{n-1} a^i d^{n-1-i}) \\ 0 & d^n \end{pmatrix}$$

gilt. Die Summe $\sum_{i=0}^{n-1} a^i d^{n-1-i}$ lässt sich mittels der geometrischen Reihe vereinfachen:

$$\begin{aligned} \sum_{i=0}^{n-1} a^i d^{n-1-i} &= d^{n-1} \sum_{i=0}^{n-1} a^i d^{-i} \\ &= d^{n-1} \sum_{i=0}^{n-1} (\frac{a}{d})^i \\ &= d^{n-1} \cdot \frac{1 - (\frac{a}{d})^n}{1 - \frac{a}{d}} \end{aligned}$$

Da $ord(a) = ord(d) = q$ und $a \neq 1$ oder $d \neq 1$ ist, gilt für $1 \leq n < q$, dass $a^n \neq 1$ oder $d^n \neq 1$ und damit $M^n \neq I$. Für $n = q$ hat $1 - (\frac{a}{d})^n$ den Wert Null und $M^q = I$. Zusammen folgt, dass $ord(M) = q$.

Sei g und $h = g^x \pmod p$ eine Instanz des DLP in G_q und $\mathcal{A}$ ein Algorithmus, der das DLP für Instanzen der Form (M_1, M_2) mit $M_1 \in \mathcal{M}$ und $M_2 \in < M_1 >$ löst. Der folgende Algorithmus $\mathcal{A}^*$ löst dann das DLP in G_q:

1. Wähle zufällig einen Wert y mit $1 < y \leq q-1$.
2. Wähle $b \in_R \mathbb{Z}_p$ zufällig.
3. Setze $M_1 = \begin{pmatrix} g & b \\ 0 & g^y \end{pmatrix}$ und $M_2 = \begin{pmatrix} h & b(h^y g^{-y} \frac{1-\frac{h}{h^y}}{1-\frac{g}{g^y}}) \\ 0 & h^y \end{pmatrix}$.
4. Gib (M_1, M_2) als Eingabe an den Algorithmus $\mathcal{A}$.
5. Gib den selben Wert aus wie Algorithmus $\mathcal{A}$.

Da $b(h^y g^{-y}(1-\frac{h}{h^y})(1-\frac{g}{g^y})^{-1}) = b((g^y)^{x-1}(1-(\frac{g}{g^y})^x)(1-\frac{g}{g^y})^{-1})$ ist und $x = \log_g h = \log_{g^y} h^y$ gilt, ist $x = \log_{M_1} M_2$, d.h. Algorithmus $\mathcal{A}^*$ gibt den gesuchten diskreten Logarithmus $x = \log_g h = \log_{M_1} M_2$ aus.

Seien nun mit $M_1, M_2 \in \mathcal{M}$ wobei $M_2 \in <M_1>$ eine Instanz des DLP in $\mathcal{M}$ gegeben. Seien $m_1, m_2 \in G_q$ die (1,1)-Einträge von M_1 bzw. M_2. Dabei kann o.B.d.A. angenommen werden, dass $m_1 \neq 1$ gilt. Andernfalls betrachtet man die (2,2)-Einträge. Dann gilt $\log_{M_1} M_2 = \log_{m_1} m_2$. Ein Algorithmus, der diskrete Logarithmen in G_q berechnet, liefert also auch den gesuchten diskreten Logarithmus in $\mathcal{M}$. □

2.7.3 Zentralisatoren in linearen Gruppen

Sei $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \in GL(2, \mathbb{Z}_p)$ gegeben. Dieser Abschnitt beschäftigt sich mit Eigenschaften von Elementen $\widehat{X} = \begin{pmatrix} \widehat{x}_1 & \widehat{x}_2 \\ \widehat{x}_3 & \widehat{x}_4 \end{pmatrix} \in GL(2, \mathbb{Z}_p)$ aus dem Zentralisator von X, d.h. für die $X \cdot \widehat{X} = \widehat{X} \cdot X$ gilt.

Aus der Gleichung $X \cdot \widehat{X} = \widehat{X} \cdot X$ erhält man durch Koeffizientenvergleich zunächst folgendes lineares Gleichungssystem:

$$\begin{array}{rcrcrcrcl} & - & x_3 \cdot \widehat{x}_2 & + & x_2 \cdot \widehat{x}_3 & & & = 0 & (I) \\ -x_2 \cdot \widehat{x}_1 & + & (x_1 - x_4) \cdot \widehat{x}_2 & & & + & x_2 \cdot \widehat{x}_4 & = 0 & (II) \\ x_3 \cdot \widehat{x}_1 & & & + & (x_4 - x_1) \cdot \widehat{x}_3 & - & x_3 \cdot \widehat{x}_4 & = 0 & (III) \end{array}$$

Für $x_3 \neq 0$ reduziert sich wegen $x_2 \cdot (III) + x_3 \cdot (II) - (x_4 - x_1) \cdot (I) = 0$ das Gleichungssystem auf ein System mit nur 2 linear unabhängigen Gleichungen. In diesem Fall sind die Zentralisatorelemente $\widehat{X} \in Z(X)$ von der Form

$$\begin{pmatrix} \frac{x_1-x_4}{x_3} \cdot \widehat{x}_3 + \widehat{x}_4 & \frac{x_2}{x_3} \cdot \widehat{x}_3 \\ \widehat{x}_3 & \widehat{x}_4 \end{pmatrix} = \begin{pmatrix} \frac{x_1-x_4}{x_3} \cdot \widehat{x}_3 & \frac{x_2}{x_3} \cdot \widehat{x}_3 \\ \widehat{x}_3 & 0 \end{pmatrix} + \widehat{x}_4 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

für beliebige $\widehat{x}_3, \widehat{x}_4 \in \mathbb{Z}_p$ mit $(\widehat{x}_3, \widehat{x}_4) \neq (0, 0)$.

Satz 9 *Seien* $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}, \widehat{X} = \begin{pmatrix} \widehat{x}_1 & \widehat{x}_2 \\ \widehat{x}_3 & \widehat{x}_4 \end{pmatrix} \in GL(2, \mathbb{Z}_p)$ *mit* $X \notin Z(GL(2, \mathbb{Z}_p))$. *Dann gilt* $\widehat{X} \in Z(X)$ *genau dann, wenn es ein* $c \in \mathbb{Z}_p$ *gibt mit i)* $c \cdot x_3 = \widehat{x}_3$, *ii)* $c \cdot x_2 = \widehat{x}_2$ *und iii)* $c \cdot (x_1 - x_4) = \widehat{x}_1 - \widehat{x}_4$.

Beweis: Wir unterscheiden die Fälle $x_3 = 0$ und $x_3 \neq 0$.
1. Fall $x_3 \neq 0$: Sei $\widehat{X} \in Z(X)$. Dann gelten Gleichungen (I)-(III) des Gleichungssystems. Sei ferner zunächst $\widehat{x}_3 \neq 0$. Setze $c := \frac{\widehat{x}_3}{x_3}$. Aus Gleichung (I) folgt dann $\widehat{x}_2 \cdot x_3 = x_2 \cdot \widehat{x}_3 = x_2 \cdot x_3 \cdot c$. Mit $x_3 \neq 0$ folgt $c \cdot x_2 = \widehat{x}_2$. Gleichung (III) liefert $x_3 \cdot (\widehat{x}_1 - \widehat{x}_4) = \widehat{x}_3 \cdot (x_1 - x_4) = c \cdot x_3 \cdot (x_1 - x_4)$. Mit $x_3 \neq 0$ folgt $\widehat{x}_1 - \widehat{x}_4 = c \cdot (x_1 - x_4)$.
Sei jetzt $\widehat{x}_3 = 0$. Dann folgt aus (I)-(III), dass $\widehat{x}_3 = \widehat{x}_2 = \widehat{x}_1 - \widehat{x}_4 = 0$ gilt. Der Wert $c = 0$ erfüllt die geforderten Bedingungen.
Seien nun die Bedingungen i) bis iii) erfüllt. Wir zeigen, dass dann die Gleichungen (I) und (II) und damit $\widehat{X} \in Z(X)$ gelten. Wegen $x_2 \cdot \widehat{x}_3 = c \cdot x_2 \cdot x_3 = \widehat{x}_2 \cdot x_3$ ist Gleichung (I) und wegen $\widehat{x}_2 \cdot (x_1 - x_4) = c \cdot x_2 \cdot (x_1 - x_4) = x_2 \cdot (\widehat{x}_1 - \widehat{x}_4)$ ist auch Gleichung (II) erfüllt.
2. Fall $x_3 = 0$: Da $X \notin Z(GL(2, \mathbb{Z}_p))$ ist, muss $x_2 \neq 0$ oder $x_1 - x_4 \neq 0$ gelten.
Sei $\widehat{X} \in Z(X)$. Aus Gleichung (I) bzw (III) folgt, dass $\widehat{x}_3 = 0$ ist. Für $x_2 \neq 0$ setze $c := \frac{\widehat{x}_2}{x_2}$. Gleichung (II) liefert $x_2 \cdot (\widehat{x}_1 - \widehat{x}_4) = \widehat{x}_2 \cdot (x_1 - x_4) = c \cdot x_2 \cdot (x_1 - x_4)$. Da $x_2 \neq 0$ ist, folgt $c \cdot (x_1 - x_4) = \widehat{x}_1 - \widehat{x}_4$. Für $x_1 - x_4 \neq 0$ setze $c := \frac{\widehat{x}_1 - \widehat{x}_4}{x_1 - x_4}$. Gleichung (II) liefert $\widehat{x}_2 \cdot (x_1 - x_4) = c \cdot x_2 \cdot (x_1 - x_4) = c \cdot x_2 \cdot (x_1 - x_4)$. Da $x_1 - x_4 \neq 0$ ist, folgt $\widehat{x}_2 = c \cdot x_2$. In beiden Fällen sind also die Bedingungen i) bis iii) erfüllt.
Seien i) bis iii) erfüllt. Aus $x_3 = 0$ und i) folgt, dass $\widehat{x}_3 = 0$. Damit sind die Gleichungen (I) und (III) erfüllt. Aus ii) und iii) folgt, dass auch Gleichung (II) erfüllt ist. □

2.7.4 Die Konjugationsprobleme in den linearen Gruppen

Seien $X, Y, Z \in GL(2, \mathbb{Z}_p)$ mit $Y = Z \cdot X \cdot Z^{-1}$. Dann ist X, Y eine Instanz des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$. Um diese zu lösen, betrachten wir die zu $Y = Z \cdot X \cdot Z^{-1}$ äquivalente Gleichung $Y \cdot Z = Z \cdot X$.

Für $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$, $Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}$ und $Z = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix}$ ergibt sich daraus das folgende lineare Gleichungssystem:

$$\begin{array}{rcrcrcrcll}
(x_1 - y_1) \cdot z_1 & + & x_3 \cdot z_2 & - & y_2 \cdot z_3 & & & = 0 & (I) \\
x_2 \cdot z_1 & + & (x_4 - y_1) \cdot z_2 & & & - & y_2 \cdot z_4 & = 0 & (II) \\
-y_3 \cdot z_1 & & & + & (x_1 - y_4) \cdot z_3 & + & x_3 \cdot z_4 & = 0 & (III) \\
& - & y_3 \cdot z_2 & + & x_2 \cdot z_3 & + & (x_4 - y_4) \cdot z_4 & = 0 & (IV)
\end{array}$$

Nutzt man aus, dass die Spur und die Determinante invariant unter Konjugation sind (siehe Satz 20 auf Seite 55), so gilt für $y_3 \neq 0$:

$$\begin{array}{rcrcrcrl}
y_3 \cdot (I) & + & & & (x_1 - y_1) \cdot (III) & + & x_3 \cdot (IV) & = 0 \\
& & y_3 \cdot (II) & + & x_2 \cdot (III) & + & (x_4 - y_1) \cdot (IV) & = 0
\end{array}$$

Das betrachtete Gleichungssystem ist also äquivalent zu folgendem Gleichungssystem bestehend aus 2 unabhängigen linearen Gleichungen:

$$\begin{array}{rcrcrcl} -y_3 \cdot z_1 & & & + & (x_1 - y_4) \cdot z_3 & + & x_3 \cdot z_4 & = 0 \\ & - & y_3 \cdot z_2 & + & x_2 \cdot z_3 & + & (x_4 - y_4) \cdot z_4 & = 0 \end{array}$$

Als Lösungsmenge der Instanz X, Y des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ ergibt sich damit:

$$L = \left\{ \begin{pmatrix} \frac{x_1 - y_4}{y_3} z_3 + \frac{x_3}{y_3} z_4 & \frac{x_2}{y_3} z_3 + \frac{x_4 - y_4}{y_3} z_4 \\ z_3 & z_4 \end{pmatrix} \;\middle|\; z_3, z_4 \in \mathbb{Z}_p \text{ nicht beide gleich Null} \right\}$$

Ist eine Lösung aus $SL(2, \mathbb{Z}_p)$ gesucht, d.h. $\det(Z) = 1$, so kann z_3 durch einen Term ersetzt werden, der nur von z_4 abhängt.

Das spezielle Konjugationsproblem: Wir zeigen nun, wie man in endlich erzeugten Gruppen eine Lösung des speziellen Konjugationsproblems als simultane Lösung geeigneter Instanzen des Konjugationsproblems erhält. Dieses allgemeine Konstruktionsprinzip wenden wir anschließend auf die endlich erzeugten Gruppen $GL(2, \mathbb{Z}_p)$ und $SL(2, \mathbb{Z}_p)$ an und zeigen wie man in diesen Gruppen simultane Lösungen des Konjugationsproblems berechnen kann. Dazu benutzen wir das folgende Lemma.

Lemma 1 *Sei G eine Gruppe und seien $g_1, \ldots, g_n \in G$. Dann gilt*

$$\bigcap_{i=1}^{n} Z(g_i) = Z(< g_1, \ldots, g_n >).$$

Beweis: Seien $g \in \bigcap_{i=1}^{n} Z(g_i)$ und $h \in < g_1, \ldots g_n >$ beliebig gewählt. Nach Wahl von g gilt $g \in Z(g_i)$ für $1 \leq i \leq n$. Da sich h als Produkt der $g_1, \ldots, g_n$ darstellen lässt, folgt $g \in Z(h)$. Weil $h \in < g_1, \ldots g_n >$ beliebig gewählt wurde, folgt ferner $g \in Z(< g_1, \ldots, g_n >)$ und damit $\bigcap_{i=1}^{n} Z(g_i) \subseteq Z(< g_1, \ldots, g_n >)$.
Sei nun $h \in Z(< g_1, \ldots, g_n >)$ beliebig gewählt. Da insbesondere $g_1, \ldots, g_n \in < g_1, \ldots, g_n >$ gilt, folgt $h \in Z(g_i)$ für $1 \leq i \leq n$ und damit $Z(< g_1, \ldots, g_n >) \subseteq \bigcap_{i=1}^{n} Z(g_i)$. □

Sei $\{g_1, \ldots, g_n\}$ ein Erzeugendensystem der Gruppe G. Sei ferner ein innerer Automorphismus $Inn(g) : G \to G$ gegeben. Durch Berechnung von $Inn(g)(g_i)$ für $1 \leq i \leq n$ erhält man n Instanzen $g_i, Inn(g)(g_i)$, $1 \leq i \leq n$, des Konjugationsproblems in G. Für diese Instanzen existiert eine simultane Lösung, d.h. es gibt ein $h \in G$ mit $hg_ih^{-1} = Inn(g)(g_i)$ für $1 \leq i \leq n$. Diese simultane Lösung ist auch eine Lösung der Instanz $Inn(g)$ des speziellen Konjugationsproblems in G, d.h. es gilt $Inn(g) = Inn(h)$:
Aus $hg_ih^{-1} = Inn(g)(g_i)$ für $1 \leq i \leq n$ folgt, dass h von der Form $h = g \cdot z_i$ mit $z_i \in Z(g_i)$ sein muss. Mit dem Lemma folgt also, dass $z_i \in \bigcap_{i=1}^{n} Z(g_i) = Z(< g_1, \ldots, g_n >) = Z(G)$ sein muss.

Wie wir bereits wissen, wird die Gruppe $SL(2,\mathbb{Z}_p)$ von den beiden Matrizen $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ und $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ erzeugt. Um das spezielle Konjugationsproblem in $SL(2,\mathbb{Z}_p)$ zu lösen, reicht es also, zwei Instanzen des Konjugationsproblems in $SL(2,\mathbb{Z}_p)$ simultan zu lösen.

Wir werden jetzt untersuchen, welche Bedingungen Instanzen des Konjugationsproblem in $GL(2,\mathbb{Z}_p)$ erfüllen müssen, damit deren simultane Lösung eine Lösung des speziellen Konjugationsproblem liefert.

Simultane Lösungen von Instanzen des Konjugationsproblems: Seien mit $X, Y = ZXZ^{-1}$ und $\widehat{X}, \widehat{Y} = Z\widehat{X}Z^{-1}$ für $X, Y, \widehat{X}, \widehat{Y}, Z \in GL(2,\mathbb{Z}_p)$ zwei Instanzen des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$ gegeben. Gesucht ist eine simultane Lösung dieser beider Instanzen, d.h. ein Element $\widehat{Z} \in GL(2,\mathbb{Z}_p)$ mit $Y = \widehat{Z}X\widehat{Z}^{-1}$ und $\widehat{Y} = \widehat{Z}\widehat{X}\widehat{Z}^{-1}$.

Für $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$, $\widehat{X} = \begin{pmatrix} \widehat{x}_1 & \widehat{x}_2 \\ \widehat{x}_3 & \widehat{x}_4 \end{pmatrix}$, $Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}$, $\widehat{Y} = \begin{pmatrix} \widehat{y}_1 & \widehat{y}_2 \\ \widehat{y}_3 & \widehat{y}_4 \end{pmatrix}$ und $Z = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix}$ erhält man die Lösungen der Instanzen X,Y und $\hat{X}, \hat{Y}$ des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$ als Lösungen des folgenden linearen Gleichungssystems:

$$\begin{array}{rrcrcrcll}
-y_3 \cdot z_1 & & + & (x_1 - y_4) \cdot z_3 & + & x_3 \cdot z_4 & = 0 & (I) \\
 & -y_3 \cdot z_2 & + & x_2 \cdot z_3 & + & (x_4 - y_4) \cdot z_4 & = 0 & (II) \\
-\widehat{y}_3 \cdot z_1 & & + & (\widehat{x}_1 - \widehat{y}_4) \cdot z_3 & + & \widehat{x}_3 \cdot z_4 & = 0 & (III) \\
 & -\widehat{y}_3 \cdot z_2 & + & \widehat{x}_2 \cdot z_3 & + & (\widehat{x}_4 - \widehat{y}_4) \cdot z_4 & = 0 & (IV)
\end{array}$$

Wir wollen nun den Fall untersuchen, dass die Gleichungen (III) und (IV), die die zweite Instanz $\widehat{X}, \widehat{Y} = Z\widehat{X}Z^{-1}$ beschreiben, äquivalent sind zu den Gleichungen (I) und (II), die die erste Instanz $X, Y = ZXZ^{-1}$ beschreiben. In diesem Fall ist die Menge der Lösungen von $X, Y = ZXZ^{-1}$ gleich der Menge der simultanen Lösungen von $\widehat{X}, \widehat{Y} = Z\widehat{X}Z^{-1}$ und $X, Y = ZXZ^{-1}$ und die Betrachtung der zweiten Instanz liefert keinen zusätzlichen Beitrag zur Lösung des speziellen Konjugationsproblems.

Angenommen also, die Gleichungen (III) und (IV) ließen sich aus den Gleichungen (I) und (II) herleiten. Dann muss es Elemente $c, d \in \mathbb{Z}_p$ mit $c \cdot (I) = (III)$ und $d \cdot (II) = (IV)$ geben. Es folgt insbesondere, dass $c \cdot y_3 = \widehat{y}_3$ und $d \cdot y_3 = \widehat{y}_3$ und damit, dass $c = d$ gilt. Ferner folgt, dass $c \cdot x_2 = \widehat{x}_2$, $c \cdot x_3 = \widehat{x}_3$, $c \cdot (x_1 - y_4) = \widehat{x}_1 - \widehat{y}_4$ (V) und $c \cdot (x_4 - y_4) = \widehat{x}_4 - \widehat{y}_4$ (VI) erfüllt sein müssen. Durch Bildung der Differenz $(V) - (VI)$ erhält man die Gleichung $c \cdot (x_1 - x_4) = \widehat{x}_1 - \widehat{x}_4$. Mit Satz 9 folgt, dass $\widehat{X} \in Z(X)$ gilt.

Die Gegenrichtung gilt ebenfalls. Sei $\widehat{X} \in Z(X)$. Mit Satz 9 folgt dann, dass ein $c \in \mathbb{Z}_p$ existiert mit $c = \frac{\widehat{x}_3}{x_3} = \frac{\widehat{x}_2}{x_2} = \frac{\widehat{x}_1 - \widehat{x}_4}{x_1 - x_4}$. Damit erhält man

$$\begin{aligned}
\widehat{y}_3 &= \frac{1}{\det(Z)} \cdot (\widehat{x}_3 \cdot z_4^2 - \widehat{x}_2 \cdot z_3^2 + (\widehat{x}_1 - \widehat{x}_4) \cdot z_3 z_4) \\
&= c \cdot \frac{1}{\det(Z)} \cdot (x_3 \cdot z_4^2 - x_2 \cdot z_3^2 + (x_1 - x_4) \cdot z_3 z_4) \\
&= c \cdot y_3,
\end{aligned}$$

und

$$\begin{aligned}
\widehat{x}_1 - \widehat{y}_4 &= \widehat{x}_1 - \frac{1}{\det(Z)}(\widehat{x}_2 z_1 z_3 + \widehat{x}_4 z_1 z_4 - \widehat{x}_1 z_2 z_3 - \widehat{x}_3 z_2 z_4) \\
&= \frac{1}{\det(Z)}(\widehat{x}_1(z_2 z_3 - \det(Z)) + \widehat{x}_3 z_2 z_4 - \widehat{x}_2 z_1 z_3 - \widehat{x}_4 z_1 z_4) \\
&= \frac{1}{\det(Z)}((\widehat{x}_1 - \widehat{x}_4) z_1 z_4 + \widehat{x}_3 z_2 z_4 - \widehat{x}_2 z_1 z_3) \\
&= c \cdot \frac{1}{\det(Z)}((x_1 - x_4) z_1 z_4 + x_3 z_2 z_4 - x_2 z_1 z_3) \\
&= c \cdot (x_1 - y_4)
\end{aligned}$$

Vollkommen analog zeigt man, dass $\widehat{x}_4 - \widehat{y}_4 = c \cdot (x_4 - y_4)$ gilt. Insgesamt gilt also $c \cdot (I) = (III)$ und $c \cdot (II) = (IV)$.

Damit wurde der folgende Satz bewiesen.

Satz 10 *Seien $X, \widehat{X}, Z \in GL(2, \mathbb{Z}_p)$ mit $X, \widehat{X} \notin Z(GL(2, \mathbb{Z}_p))$. Seien ferner L und $\widehat{L}$ die Lösungsmengen der Instanzen X, ZXZ^{-1} bzw. $\widehat{X}, Z\widehat{X}Z^{-1}$ des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$. Dann gilt*

$$L = \widehat{L} \;\Leftrightarrow\; \widehat{X} \in Z(X).$$

Wir betrachten nun den Fall zweier simultaner Instanzen X, ZXZ^{-1} und $\widehat{X}, Z\widehat{X}Z^{-1}$ des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ mit $X \notin Z(\widehat{X})$. Die Lösungsmenge dieser simultanen Instanzen wird, wie wir schon gesehen haben, durch folgendes Gleichungssystem beschrieben:

$$\begin{array}{rcrcrcl}
-y_3 \cdot z_1 & & + & (x_1 - y_4) \cdot z_3 & + & x_3 \cdot z_4 & = 0 \quad (I) \\
& -y_3 \cdot z_2 & + & x_2 \cdot z_3 & + & (x_4 - y_4) \cdot z_4 & = 0 \quad (II) \\
-\widehat{y}_3 \cdot z_1 & & + & (\widehat{x}_1 - \widehat{y}_4) \cdot z_3 & + & \widehat{x}_3 \cdot z_4 & = 0 \quad (III) \\
& -\widehat{y}_3 \cdot z_2 & + & \widehat{x}_2 \cdot z_3 & + & (\widehat{x}_4 - \widehat{y}_4) \cdot z_4 & = 0 \quad (IV)
\end{array}$$

Wegen $(y_3(\widehat{x}_1 - \widehat{y}_4) - \widehat{y}_3(x_1 - y_4))(y_3(IV) - \widehat{y}_3(II)) - (y_3\widehat{x}_2 - \widehat{y}_3 x_2)(y_3(III) - \widehat{y}_3(I)) = 0$ fällt eine Gleichung weg und das Gleichungssystem vereinfacht sich zu:[5]

$$\begin{array}{ccccc} z_1 & & + & \frac{\widehat{x}_3(y_4-x_1)-x_3(\widehat{y}_4-\widehat{x}_1)}{y_3(\widehat{y}_4-\widehat{x}_1)-\widehat{y}_3(y_4-x_1)} \cdot z_4 & = 0 \\ & z_2 & + & \frac{y_4-x_4}{y_3} + \frac{\widehat{y}_3 x_2 x_3 - y_3 x_2 \widehat{x}_3}{y_3(y_3(\widehat{y}_4-\widehat{x}_1)-\widehat{y}_3(y_4-x_1))} \cdot z_4 & = 0 \\ & z_3 & + & \frac{\widehat{y}_3 x_3 - y_3 \widehat{x}_3}{y_3(\widehat{y}_4-\widehat{x}_1)-\widehat{y}_3(y_4-x_1)} \cdot z_4 & = 0 \end{array}$$

Die simultane Lösungsmenge L der beiden Instanzen X, ZXZ^{-1} und $\widehat{X}, Z\widehat{X}Z^{-1}$ ist also ein 1-dimensionaler Untervektorraum des Vektorraumes $M(2, \mathbb{Z}_p)$. Es gilt sicherlich $Z \in L$. Alle anderen Lösungen $\widehat{Z} \in L$ lassen sich darstellen als

$$\widehat{Z} = Z \cdot (c \cdot I) = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix} \cdot \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix} = \begin{pmatrix} c \cdot z_1 & c \cdot z_2 \\ c \cdot z_3 & c \cdot z_4 \end{pmatrix}$$

mit $c \in \mathbb{Z}_p$. Die Lösungsmenge lässt sich auch darstellen als $L = Z \cdot Z(GL(2, \mathbb{Z}_p))$. Dies zeigt zum einen, dass zwei Instanzen bereits ausreichen, um das spezielle Konjugationsproblem zu lösen. Betrachtet man zum anderen die Lösungsmengen $Z \cdot Z(X)$ und $Z \cdot Z(\widehat{X})$ der Instanzen X, ZXZ^{-1} bzw. $\widehat{X}, Z\widehat{X}Z^{-1}$, so folgt, dass $Z \cdot Z(X) \cap Z \cdot Z(\widehat{X}) = Z \cdot Z(GL(2, \mathbb{Z}_p))$ und damit dass $Z(X) \cap Z(\widehat{X}) = Z(GL(2, \mathbb{Z}_p))$ gilt. Zusammen mit Lemma 1 folgt, dass $Z(< X, \widehat{X} >) = Z(X) \cap Z(\widehat{X}) = Z(GL(2, \mathbb{Z}_p))$ ist.

2.8 Das DLP in $Inn(GL(2, \mathbb{Z}_p))$

In diesem Abschnitt zeigen wir, wie das DLP in $Inn(GL(2, \mathbb{Z}_p))$ auf das DLP in $GL(4, \mathbb{Z}_p)$ reduziert werden kann. Ein Angreifer, der diskrete Logarithmen in $\mathbb{Z}_p^*$ berechnen kann, könnte über die Determinante also Informationen über den gesuchten diskreten Logarithmus berechnen. Wir zeigen, wie das definierende Element g gewählt werden muss, damit dieser Angriff nicht funktioniert.

Wir nutzen aus, dass sich $Inn(g)$ für $g \in GL(2, \mathbb{Z}_p)$ zu einer linearen Abbildung

$$\begin{array}{rcl} \varphi : M(2, \mathbb{Z}_p) & \rightarrow & M(2, \mathbb{Z}_p) \\ M & \mapsto & g \cdot M \cdot g^{-1} \end{array}$$

auf dem Matrixring $M(2, \mathbb{Z}_p)$ fortsetzen lässt. Da für alle Elemente $h \in GL(2, \mathbb{Z}_p)$ die Bilder $\varphi(h) = Inn(g)(h)$ bekannt sind, können durch Ausnutzung der Linearität von φ die Bilder $\varphi(\delta_{ij})$ für $1 \leq i, j \leq 2$ berechnet werden.[6] Dadurch ist die Abbildung φ bereits

[5] Wir nehmen o.B.d.A. an, dass $y_3, \widehat{y}_3 \neq 0$ gilt. Der Fall, dass $y_3 = 0$ oder $\hat{y}_3 = 0$ ist, kann analog behandelt werden.

[6] Seien $B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, C = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \in SL(2, \mathbb{Z}_p)$. Dann lassen sich die δ_{ij} darstellen als $\delta_{11} = C - B$, $\delta_{12} = \delta_{11} \cdot C - \delta_{11}$, $\delta_{21} = C \cdot \delta_{11} - \delta 11$ und $\delta_{22} = B^2 - \delta_{11}$.

vollständig bestimmt. Da der Ring $M(2, \mathbb{Z}_p)$ ferner ein Vektorraum der Dimension 4 über $\mathbb{Z}_p$ ist, kann die Abbildung φ durch eine Matrix $M \in GL(4, \mathbb{Z}_p)$ beschrieben werden.[7]

Sei nun mit $Inn(g), Inn(g^a)$ für $g \in GL(2, \mathbb{Z}_p)$ und $1 < a < \text{ord}(g)$ eine Instanz des DLP in $Inn(GL(2, \mathbb{Z}_p))$ gegeben. Seien $M, \widehat{M} \in GL(4, \mathbb{Z}_p)$ die Matrizen, die die auf $M(2, \mathbb{Z}_p)$ fortgesetzten Abbildungen $Inn(g)$ bzw. $Inn(g^a)$ beschreiben. Dann gilt $\widehat{M} = M^a$. Es gilt sogar, dass $Inn(g)$ in $Inn(GL(2, \mathbb{Z}_p))$ und M in $GL(4, \mathbb{Z}_p)$ die selbe Ordnung haben. Damit gilt der folgende Satz.

Satz 11 *Es gilt DLP in $Inn(GL(2, \mathbb{Z}_p)) \leq_P$ DLP in $GL(4, \mathbb{Z}_p)$.*

Betrachtet man nun die Determinanten von M und $\widehat{M}$, so folgt mit dem Determinantenmultiplikationssatz, dass $\det(\widehat{M}) = (\det(M))^a$ gilt. Ein Angreifer, der diskrete Logarithmen in $\mathbb{Z}_p^*$ berechnen kann, wäre also auch in der Lage, den diskreten Logarithmus a von $Inn(g^a)$ zur Basis $Inn(g)$ modulo der Ordnung von $\det(M)$ zu berechnen.

Durch geschickte Wahl von $g \in GL(2, \mathbb{Z}_p)$ kann man erreichen, dass $\det(M) = 1$ ist und damit obigen Angriff verhindern:
Sei q eine hinreichend große Primzahl, die $p + 1$ teilt (insbesondere $q \neq 2$). Man wählt nun $g \in GL(2, \mathbb{Z}_p)$ so, dass $\text{ord}(g) = q$ ist. Die Existenz solcher Elemente folgt aus den Sylowschen Sätzen, da q damit auch ein Teiler der Gruppenordnung $(p-1)^2(p+1)p$ von $GL(2, \mathbb{Z}_p)$ ist.[8] Da $\text{ord}(Inn(g)) \mid \text{ord}(g)$ und $\text{ord}(g)$ prim ist, folgt $\text{ord}(Inn(g)) = q$.[9] Damit hat auch die der Abbildung $Inn(g)$ zugeordnete Matrix $M \in GL(4, \mathbb{Z}_p)$ die Ordnung q. Es gilt also $(\det(M))^q = 1 \pmod{p-1}$ und damit $\text{ord}(\det(M)) \mid q$. Da q so gewählt wurde, dass $q \mid p + 1$ gilt, folgt $\text{ord}(\det(M)) \mid p + 1$. Wegen $M \in GL(4, \mathbb{Z}_p)$ gilt $\det(M) \in \mathbb{Z}_p^*$. Damit muss $\text{ord}(\det(M))$ die Gruppenordnung $p - 1$ von $\mathbb{Z}_p^*$ teilen.
Da es keine Primzahl $q \neq 2$ geben kann, die $p - 1$ und $p + 1$ gleichzeitig teilt, gilt $\text{ord}(\det(M)) = 1$ und damit $\det(M) = 1$.

2.9 Semi-direkte Produkte von Gruppen

Wir benutzen die Konstruktion des semi-direkten Produktes von Gruppen, um aus gegebenen Gruppen weitere nicht-abelsche Gruppen zu konstruieren.

Definition 24 (Semi-direktes Produkt von Gruppen) *Seien G, H Gruppen und $\theta : H \to Aut(G)$ ein Homomorphismus, wobei $Aut(G)$ die Automorphismengruppe von G bezeichne. Dann bezeichnen wir die Menge*

$$G \times H = \{(g, h) \mid g \in G, h \in H\}$$

[7] Da die Bilder $\varphi(\delta_{ij})$ der kanonischen Basis δ_{ij} von $M(2, \mathbb{Z}_p)$ leicht berechnet werden können, lässt sich auch eine solche Darstellung effizient finden.

[8] Da $\text{ord}(SL(2, \mathbb{Z}_p)) = (p-1)(p+1)p$, gilt diese Konstruktion auch in $SL(2, \mathbb{Z}_p)$.

[9] Den Fall $\text{ord}(Inn(g)) = 1$ schließen wir aus, da sonst $Inn(g) = Inn(g^b) = Id$ für alle $b \in \mathbb{Z}$. In einem solchen Fall macht die Betrachtung diskreter Logarithmen wenig Sinn.

zusammen mit der Verknüpfung

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1\theta(h_1)(g_2), h_1h_2)$$

als das semi-direkte Produkt $G \times_\theta H$ der Gruppen G und H bzgl. der Abbildung θ.

Das semi-direkte Produkt $G \times_\theta H$ zweier Gruppen ist wieder eine Gruppe. Neutrales Element dieser Gruppe ist (e_G, e_H), wobei e_G und e_H die neutralen Elemente von G bzw. H sind. Das zu (g, h) inverse Element ist gegeben durch $(g, h)^{-1} = (\theta(h^{-1})(g^{-1}), h^{-1})$.

Die Untergruppe $G \times \{e_H\} = \{(g, e_H) \mid g \in G\} \cong G$ ist ein Normalteiler von $G \times_\theta H$:
Seien $(g_1, h_1) \in G \times_\theta H$ und $(g_2, e_H) \in G \times \{e_H\}$. Dann gilt

$$\begin{aligned}(g_1, h_1) \cdot (g_2, e_H) \cdot (g_1, h_1)^{-1} &= (g_1\theta(h_1)(g_2), h_1) \cdot (\theta(h_1^{-1})(g_1^{-1}), h_1^{-1}) \\ &= (g_1 \cdot \theta(h_1)(g_2) \cdot g_1^{-1}, e_H) \\ &\in G \times \{e_H\}.\end{aligned}$$

Es folgt, dass $g \cdot (G \times \{e_H\}) \cdot g^{-1} \subseteq (G \times \{e_H\})$ für alle $g \in G \times_\theta H$ und damit die Behauptung.

Wenn $\theta(H) \neq \{id\}$ gilt, so ist $G \times_\theta H$ nicht-abelsch, selbst wenn die Gruppen G und H beide abelsch sind:
Ist H nicht-abelsch, so existieren $h_1, h_2 \in H$ mit $h_1 \cdot h_2 \neq h_2 \cdot h_1$. Für beliebige $g_1, g_2 \in G$ gilt dann $(g_1, h_1) \cdot (g_2, h_2) \neq (g_2, h_2) \cdot (g_1, h_1)$.
Ist G nicht-abelsch, so existieren Elemente $g_1, g_2 \in G$ mit $g_1 \cdot g_2 \neq g_2 \cdot g_1$. Es gilt dann $(g_1, e_H) \cdot (g_2, e_H) = (g_1 \cdot g_2, e_H) \neq (g_2 \cdot g_1, e_H) = (g_2, e_H) \cdot (g_1, e_H)$.
Seien nun G und H abelsche Gruppen. Aus der Voraussetzung $\theta(H) \neq \{id\}$ folgt, dass es ein $h \in H$ geben muss mit $\theta(h) \neq id$. Wähle $g \in G$ mit $\theta(h)(g) \neq g$ und schreibe g als $g = g_1 \cdot g_2$ mit $g_1, g_2 \in G$. Aus $g_1 \cdot g_2 = g \neq \theta(h)(g) = \theta(h)(g_1 \cdot g_2)$ folgt $g_1 \cdot \theta(h)(g_2^{-1}) \neq g_2^{-1} \cdot \theta(h)(g_1)$ und daraus wiederum $(g_1, h) \cdot (g_2^{-1}, h) \neq (g_2^{-1}, h) \cdot (g_1, h)$.

Beispiel: Sei G eine Gruppe. Für θ wählen wir die in Abschnitt 2.6 definierte homomorphe Abbildung Inn, die jedem Gruppenelement den durch ihn definierten inneren Automorphismus zuordnet:

$$\begin{aligned}Inn: \; G &\to Aut(G) \\ g &\mapsto Inn(g)\end{aligned}$$

Ist G eine abelsche Gruppe, so ist $Inn(g) = id$ für alle $g \in G$ und das semi-direkte Produkt $G \times_{Inn} G$ ist gleich dem direkten Produkt $G \times G$.

Um in semi-direkten Produktgruppen später besser rechnen zu können, beschäftigen wir uns jetzt mit der Konjugation und mit Potenzen von Gruppenelementen.

Konjugation von Gruppenelementen

Seien $(x,y),(a,b) \in G \times_\theta H$. Dann gilt

$$(x,y)(a,b)(x,y)^{-1} = (x\theta(y)(a)\theta(b)(x^{-1}),b).$$

Potenzen von Gruppenelementen

Sei $(g,h) \in G \times_\theta H$ und $k \in \mathbb{N}$. Dann haben die Potenzen $(g,h)^k$ folgende Form:

$$\begin{aligned}(g,h)^k &= (\prod_{i=0}^{k-1}\theta(h^i)(g), h^k) \\ &= (\psi^{k-1}(g), h^k)\,,\end{aligned}$$

wobei die Abbildung ψ definiert ist durch $x \mapsto g \cdot \theta(h)(x)$.

Wir beweisen diese Aussagen mittels vollständiger Induktion.
Sei $k = 1$. Dann gilt $(\prod_{i=0}^{0}\theta(ih)(g), h) = (g,h) = (\psi^0(g), h) = (g,h)$.
Seien die Aussagen für alle Werte kleiner n bewiesen. Für die erste Gleichung erhält man

$$\begin{aligned}(g,h)^{n+1} &= (g,h)^n \cdot (g,h) \\ &= (\prod_{i=0}^{n-1}\theta(h^i)(g), h^n) \cdot (g,h) \\ &= (\prod_{i=0}^{n-1}\theta(h^i)(g) \cdot \theta(h^n)(g), h^n \cdot h) \\ &= (\prod_{i=0}^{n}\theta(h^i)(g), h^{n+1}).\end{aligned}$$

Für die zweite Gleichung ergibt sich

$$\begin{aligned}(g,h)^{n+1} &= (g,h)\cdot(g,h)^n = (g,h)\cdot(\psi^{n-1}(g), h^n) \\ &= (g \cdot \theta(h)(\psi^{n-1}(g)), h^{n+1}) \\ &= (\psi(\psi^{n-1}(g)), h^{n+1}) \\ &= (\psi^n(g), h^{n+1}).\end{aligned}$$

Dabei wurde vorausgesetzt, dass H eine multiplikative Gruppe ist. Für additive Gruppen H lautet die analoge Aussage:

$$\begin{aligned}(g,h)^k &= (\prod_{i=0}^{k-1}\theta(ih)(g), kh) \\ &= (\psi^{k-1}(g), kh).\end{aligned}$$

3 Das MOR Kryptosystem

Das MOR Kryptosystem ist ein asymmetrisches Verschlüsselungsverfahren, das auf der Crypto 2001 von Paeng, Ha, Kim, Chee und Park vorgeschlagen wurde [42]. Es kann auf endlich erzeugten nicht-abelschen Gruppen G ausgeführt werden, in denen das Wortproblem effizient lösbar ist.
Die Sicherheit des MOR Systems basiert auf der Schwierigkeit, diskrete Logarithmen in der Gruppe der inneren Automorphismen $Inn(G)$ zu berechnen. Es hat in seinem Aufbau große Ähnlichkeiten mit dem ElGamal Kryptosystem [15]: Die Abbildung $Inn(g)$ legt als Teil des öffentlichen Schlüssels die zyklische Untergruppe $< Inn(g) >$ von $Inn(G)$ fest. Durch eine Diffie-Hellman Schlüsselvereinbarung auf $< Inn(g) >$ einigen sich Sender und Empfänger auf eine Funktion $Inn(g^{ab}) \in < Inn(g) >$. Der zu einer Nachricht $m \in G$ gehörende Geheimtext besteht neben einer Komponente $Inn(g^b)$ für die Diffie-Hellman Schlüsselvereinbarung aus dem Bild $Inn(g^{ab})(m)$ von m unter $Inn(g^{ab})$.

3.1 MOR auf endlich erzeugten nicht-abelschen Gruppen

Sei G eine endlich erzeugte nicht-abelsche Gruppe und seien $\gamma_1, \ldots, \gamma_n$ Generatoren von G. Sei ferner das Wortproblem effizient lösbar in G, d.h. für jedes $g \in G$ soll effizient eine Darstellung von g als Produkt der Generatoren γ_i, $1 \leq i \leq n$, berechenbar sein. Das MOR System besteht aus den folgenden Algorithmen.

Schlüsselerzeugung:

- Wähle ein $g \in G$ mit $g \notin Z(G)$ und $\text{ord}(g) = p$, p prim.
- Der geheime Schlüssel besteht aus einem zufällig gewählten Wert $a \in_R \mathbb{Z}_p$.
- Der öffentliche Schlüssel besteht aus den beiden Funktionen $Inn(g)$ und $Inn(g^a)$ (gegeben als $\{Inn(g)(\gamma_i)\}_{1 \leq i \leq n}$ und $\{Inn(g^a)(\gamma_i)\}_{1 \leq i \leq n}$).

Verschlüsselung:

1. Stelle die zu verschlüsselnde Nachricht $m \in G$ als Produkt der Generatoren γ_i der Gruppe G dar, d.h. $m = \gamma_{i_1} \cdot \cdots \cdot \gamma_{i_k}$ mit $i_j \in \{1, \ldots n\}$ für $1 \leq j \leq k$ für ein $k \in \mathbb{N}$.
2. Wähle $b \in_R \mathbb{Z}_p$ und berechne $(Inn(g^a))^b$, d.h. $\{(Inn(g^a))^b(\gamma_i)\}_{1 \leq i \leq n}$.
3. Berechne $E = Inn(g^{ab})(m) = (Inn(g^a))^b(m) = \prod_{j=1}^{k} (Inn(g^a))^b(\gamma_{i_j})$.
4. Berechne $\phi = Inn(g)^b$, d.h. $\{Inn(g^b)(\gamma_i)\}_{1 \leq i \leq n}$.
5. Das Paar (E, ϕ) ist der Geheimtext des Klartextes m.

Entschlüsselung:

1. Stelle die erste Komponente E des Geheimtextes als Produkt der Generatoren γ_i der Gruppe G dar, d.h. $E = \gamma_{i_1} \cdot \cdots \cdot \gamma_{i_l}$ mit $i_j \in \{1, \ldots n\}$ für $1 \leq j \leq l$ für ein $l \in \mathbb{N}$.
2. Berechne ϕ^{-a}, d.h. $\{\phi^{-a}(\gamma_i)\}_{1 \leq i \leq n}$.
3. Berechne $\phi^{-a}(E) = \prod_{j=1}^{l} \phi^{-a}(\gamma i_j)$.

Durch die Primalität der Ordnung von g ist sichergestellt, dass mit $m \notin Z(g)$ auch immer $m \notin Z(g^k)$ gilt für alle $k \not\equiv 0 \pmod{\text{ord}(g)}$. Damit ist $E = Inn(g^{ab})(m) \neq m$ für alle $m \in G$ mit $m \notin Z(g)$.
Von den bekannten Algorithmen zur Berechnung diskreter Logarithmen lassen sich nur die generischen Algorithmen in $< Inn(g) >$ anwenden. Es gilt $\text{ord}(g) = \text{ord}(Inn(g))$ (siehe Lemma 3 auf Seite 46). Die benutzte Primzahl p legt damit die Größe der benutzten zyklischen Gruppe $< Inn(g) >$ fest und sollte so groß gewählt werden, dass eine Berechnung diskreter Logarithmen in $< Inn(g) >$ mittels generischer Algorithmen praktisch nicht durchführbar ist.

Effizienz: Der rechenaufwändigste Teil von Ver-und Entschlüsselung ist die Berechnung von Potenzen von Elementen aus $Inn(G)$, also von $Inn(g^b)$ und $Inn(g^{ab})$ während der Verschlüsselung und ϕ^{-a} während der Entschlüsselung. Verwendet man einen Square-and-Mulitply Algorithmus zur Berechnung dieser Potenzen, müssen für zufällige Exponenten a im Mittel $1,5 \cdot \log_2 a$ Multiplikationen in $Inn(G)$ berechnet werden.
In unserem Fall sind Elemente $Inn(g)$ aus $Inn(G)$ durch die Bilder $Inn(g)(\gamma_i)$ der Generatoren γ_i von G gegeben. Um zwei Abbildungen $Inn(g)$ und $Inn(h)$ für $g, h \in G$ zu verknüpfen, müssen zunächst die $Inn(h)(\gamma_i)$ für alle $1 \leq i \leq n$ als Produkt der Generatoren $Inn(h)(\gamma_i) = \prod_{j_i=1}^{k_i} \gamma_{j_i}$ für $k_i \in \mathbb{N}$ und $j_i \in \{1, \ldots, n\}$ ausgedrückt werden. Anschließend müssen dann die entsprechenden Bilder unter $Inn(g)$ noch multipliziert werden, d.h. $Inn(g \cdot h)(\gamma_i) = \prod_{j_i=1}^{k_i} Inn(g)(\gamma_{j_i})$.
Der genaue Aufwand, der zur Multiplikation zweier inneren Automorphismen nötig ist, hängt also von der Anzahl der Generatoren γ_i, von der Anzahl der Faktoren in der Darstellung von Gruppenelementen und von der Komplexität der Multiplikationen in G ab. In Abschnitt 4.1 werden für die in [42] vorgeschlagene Gruppe genauere Abschätzungen der Laufzeiten gemacht.

Um die Effizienz ihres Verfahrens zu verbessern, schlagen die Autoren in [42] vor, den Exponenten b für mehrere Verschlüsselungen fest zu lassen. Die Abbildungen $Inn(g^b)$, $Inn(g^{ab})$ und ϕ^{-a} müssen dann nur noch einmal berechnet werden und können für mehrere Verschlüsselungen zwischen dem selben Sender und Empfänger verwendet werden.
Dies hat einige Konsequenzen. Das MOR Schema ist in dieser abgewandelten Form ein deterministisches Verfahren, d.h. gleiche Klartexte werden auch immer auf gleiche Geheimtexte abgebildet. Der Geheimtext einer Nachricht m ist das Bild $Inn(g^{ab})(m)$ von m unter der Abbildung $Inn(g^{ab})$. Die Verschlüsselungsfunktion ist also insbesondere homomorph, d.h.

aus zwei Geheimtexten $Inn(g^{ab})(m_1)$ und $Inn(g^{ab})(m_2)$ kann ohne Kenntnis der zugehörigen Klartexte m_1 bzw. m_2 der Geheimtext $Inn(g^{ab})(m_1 \cdot m_2) = Inn(g^{ab})(m_1) \cdot Inn(g^{ab})(m_2)$ der Nachricht $m_1 \cdot m_2$ berechnet werden. Verwendet man als zugrundeliegende Gruppe G die spezielle lineare Gruppe $SL(2, \mathbb{Z}_p)$, so ist die Verschlüsselungsfunktion $Inn(g^{ab})$, aufgefasst als Funktion auf dem Vektorraum $M(2, \mathbb{Z}_p)$, sogar homomorph bzgl. Multiplikation und Addition.
In einem Chosen-Plaintext Angriff hätte ein Angreifer Zugriff auf ein Verschlüsselungsorakel, d.h. für beliebige $m \in G$ liefert ihm dieses Orakel die Funktionswerte $Inn(g^{ab})(m)$. Wäre der Angreifer ferner in der Lage, das spezielle Konjugationsproblem in G zu lösen, d.h. er kann ein $\hat{g} \in G$ berechnen mit $Inn(\hat{g}) = Inn(g^{ab})$, so wäre er in der Lage, alle folgenden Geheimtexte entschlüsseln zu können. Die Sicherheit des MOR Systems mit festem Exponenten b gegen Chosen-Plaintext Angriffe beruht also nicht auf der Schwierigkeit, diskrete Logarithmen in $Inn(G)$ berechnen zu können, sondern auf dem einfacheren speziellen Konjugationsproblem in G.
Um diese Schwierigkeiten zu vermeiden, wird in [42] vorgeschlagen, die zu verschlüsselnde Nachrichten m nicht aus der Gruppe G selbst sondern aus einer anderen Struktur G_1 zu wählen und dann mittels eines probabilistischen Padding-Verfahrens in G einzubetten. Das benutzte Padding-Verfahren muss also so konstruiert sein, dass die MOR-Verschlüsselung mit Padding zum einen nicht homomorph ist und ein Angreifer zum anderen aus Paaren von Klartexten und zugehörigen MOR-Geheimtexten nicht genügend Informationen erhält, um die Konjugationsprobleme in G lösen zu können. In Abschnitt 4.2 wird ein solches Padding-Verfahren für $G = SL(2, \mathbb{Z}_p)$ und das semi-direkte Produkt $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ vorgestellt.

3.1.1 Die Sicherheit des MOR Schemas

Ein naheliegender Angriff auf ein asymmetrisches Verschlüsselungsverfahren ist der Versuch, den geheimen Schlüssel eines Teilnehmers aus dem zugehörigen öffentlichen Schlüssel zu berechnen. Im Falle von MOR müsste ein Angreifer dazu diskrete Algorithmen in der von $Inn(g)$ erzeugten zyklischen Untergruppe der Gruppe $Inn(G)$ berechnen können. Der folgende Abschnitt soll, unabhängig von der konkret gewählten Gruppe, einige Anhaltspunkte über die Schwierigkeit dieses Problems geben.

Es gibt in G jeweils genau $|Z(G)|$ Elemente, die den selben inneren Automorphismus definieren (siehe Satz 4 auf Seite 23). Es wäre möglich, dass es Gruppenelemente $g, h \in G$ mit $Inn(g) = Inn(h)$ und $Inn(g^a) = Inn(h^b)$ gibt, wobei $a \not\equiv b \pmod{\mathrm{ord}(g)}$. In diesem Fall wäre der diskrete Logarithmus in $< Inn(g) >$ nicht eindeutig definiert. Der folgende Satz zeigt, dass ein solcher Fall für Gruppenelemente $g \in G$, die die in der Schlüsselerzeugung geforderten Voraussetzungen erfüllen, nicht eintreten kann.

Satz 12 *Sei $g \notin Z(G)$ und $\mathrm{ord}(g)$ prim. Dann hat die Instanz $Inn(g), Inn(g^a)$ des DLP in $< Inn(g) >$ eine eindeutige Lösung, d.h. angenommen es existieren $h \in G$ und $b \in \mathbb{Z}$ mit $Inn(g) = Inn(h)$ und $Inn(g^a) = Inn(h^b)$, so gilt $a \equiv b \pmod{\mathrm{ord}(g)}$.*

Beweis: Sei $Inn(g) = Inn(h)$ und $Inn(g^a) = Inn(h^b)$ für ein $h \in G$ und ein $b \in \mathbb{Z}$. Wegen Satz 4 auf Seite 23 folgt, dass $h = g \cdot z_1$ und $h^b = g^a \cdot z_2$ mit $z_1, z_2 \in Z(G)$ gilt. Daraus ergibt sich, dass die Gleichung $g^a \cdot z_2 = h^b = (g \cdot z_1)^b = g^b \cdot z_1^b$ erfüllt sein muss und damit $g^{b-a} \in Z(G)$ ist. Da nach Voraussetzung g so gewählt wurde, dass $g \notin Z(G)$ und $\mathrm{ord}(g)$ prim ist, folgt $a \equiv b \pmod{\mathrm{ord}(g)}$. □

Das folgende Korollar sagt etwas über den Zusammenhang des DLP in $Inn(G)$ mit dem DLP und dem SCP in der zugrundeliegenden Gruppe G aus.

Korollar 2 *Sei $g \notin Z(G)$ und $\mathrm{ord}(g)$ prim. Sei mit $Inn(g), Inn(g^a)$ eine Instanz des DLP in $Inn(G)$ gegeben. Für jede Lösung $h \in G$ der Instanz $Inn(g)$ des SCP in G gibt es mit h^a genau eine Lösung der Instanz $Inn(g^a)$ des SCP in G, die in $<h>$ liegt.*

Beweis: Wir zeigen zunächst, dass h^a eine Lösung der Instanz $Inn(g^a)$ des SCP in G ist. Aus $Inn(h) = Inn(g)$ folgt mit Satz 4 auf Seite 23, dass h von der Form $h = g \cdot z$ für ein $z \in Z(G)$ ist. Damit gilt $h^a = (g \cdot z)^a = g^a \cdot z^a$ und mit Satz 4 folgt die Gleichheit $Inn(h^a) = Inn(g^a)$.
Angenommen, es gibt eine weiteres Element $h^b \in <h>$ mit $Inn(h^a) = Inn(h^b)$. Dann folgt mit Satz 12 die Gleichheit $a \equiv b \pmod{\mathrm{ord}(g)}$. □

Es sei $\mathcal{A}$ nun ein Angreifer, der in der Lage ist, das spezielle Konjugationsproblem in G zu lösen und diskrete Logarithmen in G zu berechnen. Angreifer $\mathcal{A}$ könnte versuchen, das DLP in $Inn(G)$ zu lösen, indem er zunächst das spezielle Konjugationsproblem für $Inn(g)$ und $Inn(g^a)$ löst, d.h. Gruppenelemente $g_1, g_2 \in G$ mit $Inn(g) = Inn(g_1)$ und $Inn(g^a) = Inn(g_2)$ berechnet. Im zweiten Schritt könnte er g_1 und g_2 als Eingabe für seinen Algorithmus zur Berechnung diskreter Logarithmen in G verwenden. Für $g_1 \in <g_2>$ sagt uns Satz 12, dass er dadurch die richtige Lösung erhält. Im Falle $g_1 \notin <g_2>$ kann der DLP-Algorithmus jedoch nicht direkt eingesetzt werden.
Korollar 2 sagt aus, dass g^a die einzige Lösung der Instanz $Inn(g^a)$ des SCP aus $<g>$ ist. Liefert der Algorithmus zur Lösung des SCP in G zufällige Lösungen $g_1, g_2 \in G$, so ist die Wahrscheinlichkeit, dass $g_1 \in <g_2>$ gilt, gerade $\frac{1}{|Z(G)|}$. Durch die Wahl einer Gruppe G mit hinreichend großem Zentrum kann dieser Angriff also verhindert werden.

Das nächste Lemma beschreibt eine weitere Beziehung zwischen Gruppenelementen $g \in G$ und den durch sie definierten inneren Automorphismen $Inn(g)$.

Lemma 2 *Sei $g \notin Z(G)$ und $\mathrm{ord}(g)$ prim. Dann gilt $\mathrm{ord}(g) = \mathrm{ord}(Inn(g))$.*

Beweis: Es gilt $g^{\mathrm{ord}(g)} = 1_G$ und damit $(inn(g))^{\mathrm{ord}(g)} = Inn(g^{\mathrm{ord}(g)}) = Id_G$. Damit folgt die Beziehung $\mathrm{ord}(Inn(g)) \mid \mathrm{ord}(g)$.
Ferner gilt $Inn(g^{\mathrm{ord}(Inn(g))}) = (Inn(g))^{\mathrm{ord}(Inn(g))} = Id_G$ und damit $g^{\mathrm{ord}(Inn(g))} \in Z(G)$. Da nach Voraussetzung $g \notin Z(G)$ und $\mathrm{ord}(g)$ prim gilt, muss $\mathrm{ord}(Inn(g)) \equiv 0 \pmod{\mathrm{ord}(g)}$ sein. Daraus folgt die Beziehung $\mathrm{ord}(g) \mid \mathrm{ord}(Inn(g))$ und insgesamt die Behauptung. □

Kombiniert man die Aussagen von Satz 12 und Lemma 2, so erhält man ein nützliches Korollar.

Korollar 3 *Sei $g \notin Z(G)$ und* $\mathrm{ord}(g)$ *prim. Dann gilt* $\log_g(g^a) = \log_{Inn(g)}(Inn(g^a))$.

Mit diesen Hilfsmitteln kann man zeigen, dass bei geeigneter Wahl des Gruppenelements $g \in G$ die Berechnung diskreter Logarithmen in der von $Inn(g)$ erzeugten zyklischen Untergruppe der Gruppe der inneren Automorphismen mindestens so schwer ist, wie die Berechnung diskreter Logarithmen in der von g erzeugten zyklischen Untergruppe von G.

Satz 13 *Sei $g \notin Z(G)$ und* $\mathrm{ord}(g)$ *prim. Dann gilt*

$$DLP \text{ in } <g> \leq_P DLP \text{ in } <Inn(g)>.$$

Beweis: Sei $\mathcal{A}$ ein Algorithmus, der das DLP in $<Inn(g)>$ löst und mit g, g^a eine Instanz des DLP in $<g>$ gegeben. Da die Werte g und g^a bekannt sind, können die Funktionen $Inn(g)$ und $Inn(g^a)$ effizient berechnet werden. Auf Eingabe $(Inn(g), Inn(g^a))$ gibt Algorithmus $\mathcal{A}$ dann den gesuchten diskreten Logarithmus $a = \log_{Inn(g)}(Inn(g^a)) = \log_g(g^a)$ aus. □

Die Sicherheit des geheimen Schlüssels beruht im MOR System auf der Schwierigkeit der Berechnung diskreter Logarithmen in der von $Inn(g)$ erzeugten Untergruppe von $Inn(G)$. Wählt man $g \in G$ so, dass das DLP schwer ist in der von g erzeugten zyklischen Untergruppe von G, so ist das nach Satz 13 bereits hinreichend für die Schwierigkeit des DLP in der von $Inn(g)$ erzeugten zyklischen Untergruppe von $Inn(G)$.

In [42] wird kein formaler Sicherheitsbeweis für das MOR System gegeben. Es werden lediglich notwendige aber keine hinreichenden Bedingungen für die Sicherheit des Verfahrens angegeben. Der geheime Schlüssel kann genau dann effizient berechnet werden, wenn das Diskreter Logarithmus Problem in $<Inn(g)>$ effizient gelöst werden kann. In vielen Gruppen (auch in den in [42] und [43] vorgeschlagenen) ist die Kenntnis des geheimen Schlüssels jedoch nicht nötig, um Informationen über die verschlüsselten Klartexte zu erhalten (siehe etwa Abschnitt 4.3).

Hier sind noch einmal die notwendigen Bedingungen für nicht-abelsche Gruppen G aufgeführt, damit sie im MOR System benutzt werden können:

1. **Darstellung und Berechnung der inneren Automorphismen:** Die Gruppe G muss endlich erzeugt sein. Die inneren Automorphismen $Inn(g)$ lassen sich dann als Bilder der Generatoren γ_i, $1 \leq i \leq n$, von G darstellen. Aus Effizienzgründen sollte die Anzahl n der Generatoren nicht zu groß sein.
   Das Wortproblem muss effizient lösbar sein in G, d.h. für jedes $g \in G$ muss effizient eine Darstellung von g als Produkt der Generatoren γ_i, $1 \leq i \leq n$, berechenbar sein.

2. **Sicherheit des geheimen Schlüssels:** Für die Schlüsselerzeugung muss ein effizienter probabilistischer Algorithmus existieren, der $g \in G$ berechnet, so dass das diskrete Logarithmus Problem schwer ist in der von $Inn(g)$ erzeugten zyklischen Untergruppe von $Inn(G)$.

3. **Größe des Zentrums $Z(G)$:** Um zu verhindern, dass durch Lösen des SCP und des DLP in G diskrete Logarithmen in $< Inn(g) >$ berechnet werden können, sollte G so gewählt werden, dass das Zentrum $Z(G)$ hinreichend groß ist.
Da die Ordnung der Gruppe $Inn(G)$ über $|Inn(G)| = \frac{|G|}{|Z(G)|}$ von der Größe des Zentrums $Z(G)$ abhängt, sollte das Zentrum aber auch nicht zu groß sein.

4. **Effiziente Implementierbarkeit:** Die Speicherung der Gruppenelemente und die Durchführung der Gruppenoperationen sollte für die zugrundeliegende Gruppe G effizient möglich sein.

3.2 Das erweiterte MOR System

In [43] stellen Paeng, Kwon, Ha und Kim eine Verallgemeinerung des MOR Systems vor. Sowohl das MOR System [42] als auch das ElGamal Verschlüsselungsverfahren [15] sind Spezialfälle dieses erweiterten MOR Verfahrens.

Seien G, G' Gruppen und N ein Normalteiler von G. Sei ferner $q : G \to G/N$ die kanonische Projektion und $\varphi : G/N \to Aut(G')$ ein Homomorphismus. Insgesamt ergibt sich also folgende Sequenz

$$G \xrightarrow{q} G/N \xrightarrow{\varphi} Aut(G')$$

Das erweiterte MOR System besteht aus den folgenden Algorithmen:

Schlüsselerzeugung:

- Wähle ein $g \in G$ mit $g \notin N$ und $\text{ord}(g) = p$ mit p prim. Sei $\bar{g} = q(g)$.[10]
- Der geheime Schlüssel besteht aus eine zufällig gewählten Wert $a \in_R \mathbb{Z}_p$.
- Der öffentliche Schlüssel besteht aus den Funktionen $\varphi(\bar{g})$ und $\varphi(\bar{g})^a$, wobei $\bar{g} = q(g)$.

Verschlüsselung:

1. Wähle ein zufälliges $b \in_R \mathbb{Z}_p$ und berechne $\varphi(\bar{g})^{ab} = (\varphi(\bar{g})^a)^b$.
2. Berechne $E = \varphi(\bar{g})^{ab}(m)$. Dabei ist $m \in G'$ die zu verschlüsselnde Nachricht.
3. Berechne $\phi = \varphi(\bar{g})^b$.

[10] Es gilt, dass $\text{ord}(\bar{g}) \mid p = \text{ord}(g)$ und damit, dass $\text{ord}(\bar{g}) = p$ oder $\text{ord}(\bar{g}) = 1$. Der Fall $\text{ord}(\bar{g}) = 1$ kann nicht eintreten, da wir $g \notin N$ gefordert haben. Also gilt $\text{ord}(\bar{g}) = \text{ord}(g) = p$.

4. Das Paar (E, ϕ) ist der Geheimtext des Klartextes m.

Entschlüsselung:

1. Berechne ϕ^{-a}.
2. Berechne $m = \phi^{-a}(E)$.

Sowohl das ElGamal Verfahren als auch die Grundversion des MOR Kryptosystems lassen sich als Spezialfälle des erweiterten MOR Systems beschreiben:

1. **ElGamal Verschlüsselung**
   Sei $G = G' = \mathbb{Z}_p^*$ und $N = \{e_G\}$. Dann ist $G = G/N$ und $q = Id$. Die Sequenz vereinfacht sich damit zu

   $$G \xrightarrow{\varphi} Aut(G)$$

   Die Abbildung φ sei so definiert, dass sie jedem Element $g \in \mathbb{Z}_p^*$ den Automorphismus $Aut(g) : G \to G$, $m \mapsto m \cdot g$ zuordnet. Sei ferner g ein Generator von $\mathbb{Z}_p^*$.
   Für diese Parameter besteht der Geheimtext des erweiterten MOR Schemas also genau aus den Werten $E = \varphi(g)^{ab}(m) = m \cdot g^{ab}$ und $\phi = \varphi(g)^b = \varphi(g^b)$. Als Geheimtext der Nachricht $m \in \mathbb{Z}_p^*$ erhält man das Paar $(m \cdot g^{ab}, g^b)$ und das resultierende Verschlüsselungsverfahren entspricht dem ElGamal Verfahren, das in Abschnitt 2.5 vorgestellt wurde.

2. **Grundversion von MOR**
   Für $N = Z(G)$ und $\varphi : G/Z(G) \to Inn(G)$ erhält man das MOR Schema in der in Abschnitt 3.1 vorgestellten Grundversion.

4 MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$

In [42] zeigen die Autoren anhand von Beispielen, dass das MOR System in vielen Gruppen unsicher ist, selbst wenn sie die in Abschnitt 3.1.1 aufgeführten notwendigen Bedingungen erfüllen. Sie schlagen vor, das semi-direkte Produkt $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ als zugrundeliegende nicht-abelsche Gruppe für das MOR System zu verwenden. Im folgenden Abschnitt wird zunächst die Funktionsweise von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ beschrieben. Es schließt sich eine Analyse an, bei der die Effizienz und Sicherheit des Verfahrens untersucht werden. Dabei zeigt sich, dass auch bei Benutzung der Gruppe $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ das MOR System Schwächen aufweist, die dessen Sicherheit teils stark beeinträchtigen. Die hier vorgestellten Angriffe wurden teilweise in [54] publiziert.

Im folgenden sei

$$G = SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p\,,$$

wobei θ gegeben ist durch

$$\theta = Inn \circ \theta_1 : \mathbb{Z}_p \to Aut(SL(2,\mathbb{Z}_p))$$

mit

$$\begin{array}{rcl} \theta_1 : \mathbb{Z}_p & \to & <\alpha> \\ n & \mapsto & \alpha^n \end{array}$$

für ein $\alpha \in SL(2,\mathbb{Z}_p)$ mit Ordnung p (p prim), z.B. $\alpha = I + \delta_{12}$. Abbildung θ_1 ist ein Isomorphismus und Abbildung θ ein Homomorphismus. Insgesamt gilt $\theta(y)(x) = \theta_1(y)x\theta_1(y)^{-1}$.

Das Zentrum von G hat die Form

$$Z(G) = \{(g_1, g_2) \in G \mid g_1 = \pm\theta_1(-g_2)\}\,,$$

Ist $Inn(g)$ für ein $g = (g_1, g_2) \in G$ gegeben. So hat das spezielle Konjugationsproblem also folgende Lösungsmenge

$$L = \{(h_1, h_2) \in G \mid h_1 = \pm g_1 \cdot \theta_1(g_2 - h_2)\}$$

Die Kardinalität des Zentrums und damit der Lösungsmenge des speziellen Konjugationsproblems ist $2p$ und erfüllt damit die in Abschnitt 3.1.1 aufgestellten Anforderungen an die benutzte Gruppe G.

Wir schauen uns nun an, wie sich die Konjugation und Exponentiation in der Gruppe $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ auswirken.

Satz 14 (Konjugation) *Seien $(x,y),(a,b) \in G$. Dann gilt*

$$\begin{aligned}(x,y)(a,b)(x,y)^{-1} &= (x\theta(y)(a)\theta(b)(x^{-1}),b)\\ &= (x\theta_1(y)a\theta_1(y)^{-1}\theta_1(b)x^{-1}\theta_1(b)^{-1},b)\\ &= ((x\theta_1(y))a\theta_1(b)(x\theta_1(y))^{-1}\theta_1(b)^{-1},b).\end{aligned}$$

Beweis: Die Aussage folgt direkt durch Anwendung der Aussagen aus Abschnitt 2.9 über semi-direkte Produkte auf die Gruppe $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$. □

Eine wichtige Beobachtung an dieser Stelle ist, dass die zweite Komponente b des konjugierten Elements invariant unter Konjugation ist.

Satz 15 (Exponentiation) *Sei $(x,y) \in G$ und $n \in \mathbb{N}$. Dann gilt*

$$(x,y)^n = ((x\theta_1(y))^n\theta_1(y)^{-n}, ny).$$

Beweis: Wir zeigen die Aussage durch vollständige Induktion. Für $n = 1$ ist sie erfüllt. Angenommen, sie sei bereits bis n bewiesen. Dann gilt

$$\begin{aligned}(x,y)^{n+1} &= (x,y)^n(x,y) = ((x\theta_1(y))^n\theta_1(y)^{-n}, ny)(x,y)\\ &= ((x\theta_1(y))^n\theta_1(y)^{-n}\theta(ny)(x), (n+1)y)\\ &= ((x\theta_1(y))^n\theta_1(-ny)\theta_1(ny)x\theta_1(-ny), (n+1)y)\\ &= ((x\theta_1(y))^n x\theta_1(y)\theta_1(-(n+1)y), (n+1)y)\\ &= ((x\theta_1(y))^{n+1}\theta_1(y)^{-(n+1)}, (n+1)y).\end{aligned}$$

□

Der Exponent n taucht als Faktor in der zweiten Komponente auf. Aus den zweiten Komponenten y der Basis $g = (x,y)$ bzw. yn des Potenzwertes g^n kann der Exponent n effizient berechnet werden. Damit ist der folgende wichtige Satz bewiesen.

Satz 16 *Das diskrete Logarithmus Problem ist in $G = SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ effizient lösbar.*

Das DLP in Inn(G): Seien $(x,y),(a,b) \in G$ und $n \in \mathbb{Z}$. Mit den Sätzen 14 und 15 folgt

$$(x,y)^n(a,b)(x,y)^{-n} = ((x\theta_1(y))^n a\theta_1(b)(x\theta_1(y))^{-n}\theta_1(-b),b). \quad (1)$$

Für $g=(x,y)$ erhält man als Funktionswerte der inneren Automorphismen $Inn(g)$ und $Inn(g^n)$:

$$\begin{aligned} Inn(g)((a,b)) = (x,y)(a,b)(x,y)^{-1} &= ((x\theta_1(y))a\theta_1(b)(x\theta_1(y))^{-1}\theta_1(b)^{-1},b) \\ &\text{und} \\ Inn(g^n)((a,b)) = (x,y)^n(a,b)(x,y)^{-n} &= ((x\theta_1(y))^n a\theta_1(b)(x\theta_1(y))^{-n}\theta_1(b)^{-1},b). \end{aligned}$$

Da das Gruppenelement (a,b) sowie die Funktion θ_1 bekannt sind, können die Werte $(x\theta_1(y))a\theta_1(b)(x\theta_1(y))^{-1}$ und $(x\theta_1(y))^n a\theta_1(b)(x\theta_1(y))^{-n}$ aus den ersten Komponenten der Funktionswerte extrahiert werden. Durch die Funktionen $Inn(g), Inn(g^n) : G \to G$ sind also indirekt auch die Funktionen $Inn(x\theta_1(y)), Inn((x\theta_1(y))^n) : SL(2,\mathbb{Z}_p) \to SL(2,\mathbb{Z}_p)$ gegeben. Da das spezielle Konjugationsproblem leicht ist in $SL(2,\mathbb{Z}_p)$, können die konjugierenden Elemente $\pm x\theta_1(y)$ und $\pm(x\theta_1(y))^n$ berechnet werden (siehe Abschnitt 2.7.4).

Lemma 3 *Für $g=(x,y) \in G$ mit $\mathrm{ord}_G(g) = q$, q prim, und $g \notin Z(G)$ gilt*

$$\mathrm{ord}_G(g) \mid \mathrm{ord}_{SL(2,\mathbb{Z}_p)}(x\theta_1(y)).$$

Beweis: Sei $n = \mathrm{ord}_{SL(2,\mathbb{Z}_p)}(x\theta_1(y))$, d.h. $(x\theta_1(y))^n = I$. Dann gilt für alle $(a,b) \in G$

$$\begin{aligned} (x,y)^n(a,b)(x,y)^{-n} &= ((x\theta_1(y))^n a\theta_1(b)(x\theta_1(y))^{-n}\theta_1(-b),b) \\ &= (a,b). \end{aligned}$$

Damit gilt $Inn(g^n) = Id_G$. Folglich muss g^n im Zentrum von G liegen. Nach Wahl von g folgt, dass $g^n = 1_G$ und damit $\mathrm{ord}_G(g) \mid n$ gilt. Wegen $n = \mathrm{ord}_{SL(2,\mathbb{Z}_p)}(x\theta_1(y))$ folgt die Behauptung. □

Um eine Instanz $Inn(g), Inn(g^n)$ des DLP in $<Inn(g)>$ zu lösen, könnte ein Angreifer folgendermaßen vorgehen: Aus $Inn(g), Inn(g^n)$ berechnet er zunächst wie oben beschrieben die Werte $\pm(x\theta_1(y)), \pm(x\theta_1(y))^n \in SL(2,\mathbb{Z}_p)$. Kann der Angreifer diskrete Logarithmen in $<x\theta_1(y)>$ berechnen, so erhält er den Wert $n \bmod \mathrm{ord}(x\theta_1(y))$. Lemma 3 sagt aus, dass $\mathrm{ord}_G(g) \leq \mathrm{ord}_{SL(2,\mathbb{Z}_p)}(x\theta_1(y))$ und damit $n = n \bmod \mathrm{ord}(x\theta_1(y))$ gilt. Beim Übergang von $<Inn(g)>$ zu $<x\theta_1(y)>$ gehen also insbesondere keine Informationen über n verloren. Damit ist der folgende Satz bewiesen, der eine weitere notwendige Bedingung für die Sicherheit von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ liefert: Der Parameter $g \in G$ muss so gewählt werden, dass das DLP schwer ist in $<x\theta_1(y)>$.

Satz 17 *Sei $g=(x,y) \in G$ mit $\mathrm{ord}_G(g) = q$, q prim, und $g \notin Z(G)$. Dann gilt*

$$DLP \text{ in } <Inn(g)> \leq_P DLP \text{ in } <x\theta_1(y)>.$$

Für einen gegebenen inneren Automorphismus $Inn(g)$ gibt es im allgemeinen mehr als ein definierendes Element, nämlich alle Elemente der Form $g \cdot z$ mit $z \in Z(G)$. Der folgende Satz zeigt, dass der Wert $x\theta_1(y)$ invariant für alle Gruppenelemente ist, die den selben inneren Automorphismus definieren.

Satz 18 *Seien $g = (x_1, y_1), h = (x_2, y_2) \in G$. Dann gilt*

$$Inn(g) = Inn(h) \Leftrightarrow x_1\theta_1(y_1) = \pm x_2\theta_1(y_2).$$

Beweis: Sei zunächst $Inn(g) = Inn(h)$. Dann gibt es ein $z \in Z(G)$ mit $g = h \cdot z$ (siehe Satz 4 auf Seite 23). Da $Z(G) = \{(g_1, g_2) \in G \mid g_1 = \pm\theta_1(-g_2)\}$ muss es also ein $\hat{z} \in \mathbb{Z}_p$ geben mit

$$\begin{aligned}(x_1, y_1) &= (x_2, y_2) \cdot (\pm\theta_1(-\hat{z}), \hat{z}) \\ &= (\pm x_2\theta_1(y_2)\theta_1(-\hat{z})\theta_1(-y_2), y_2 + \hat{z}) \\ &= (\pm x_2\theta_1(-\hat{z}), y_2 + \hat{z}).\end{aligned}$$

Insgesamt folgt

$$\begin{aligned}x_1\theta_1(y_1) &= \pm x\theta_1(-\hat{z})\theta_1(y_2 + \hat{z}) \\ &= \pm x_2\theta_1(y_2).\end{aligned}$$

Sei nun $x_1\theta_1(y_1) = x_2\theta_1(y_2)$. Setze $z = (\theta_1(y_2 - y_1), y_1 - y_2) \in Z(G)$. Es folgt

$$\begin{aligned}(x_2, y_2) \cdot (\theta_1(y_2 - y_1), y_1 - y_2) &= (x_2 \cdot \theta(y_2)(\theta_1(y_2 - y_1)), y_2 + y_1 - y_2) \\ &= (x_2\theta_1(y_2)\theta_1(-y_1), y_1) \\ &= (x_1\theta_1(y_1)\theta_1(-y_1), y_1) \\ &= (x_1, y_1).\end{aligned}$$

Damit wurde gezeigt, dass ein Element $z \in Z(G)$ existiert mit $g = h \cdot z$ und damit dass $Inn(g) = Inn(h)$ gilt. Der Fall $x_1\theta_1(y_1) = -x_2\theta_1(y_2)$ verläuft vollkommen analog, wenn man $z = (-\theta_1(y_2 - y_1), y_1 - y_2) \in Z(G)$ wählt. □

Parameterwahl: Bei der Beschreibung des MOR Systems für endlich erzeugte nichtabelsche Gruppen wurde für $g \in G$ lediglich gefordert, dass g nicht im Zentrum der Gruppe G liegt und prime Ordnung hat. Dadurch ist sichergestellt, dass $E = Inn(g^{ab}(m)) \neq m$ für alle $m \in G$ mit $m \notin Z(g)$ gilt, dass $g \in G$ und $Inn(g) \in Inn(G)$ die gleiche Ordnung haben und dass das DLP in $< Inn(g) >$ eine eindeutige Lösung hat (siehe Abschnitt 3.1). Bei Benutzung von $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ als zugrundeliegende Gruppe, hat die Analyse ergeben, dass $g = (x, y) \in G$ außerdem so gewählt sein muss, dass diskrete Logarithmen in $< x\theta_1(y) >$ nicht effizient berechnet werden können, da sonst eine Berechnung des geheimen Schlüssels aus dem öffentlichen Schlüssel möglich ist.

In [42] wird vorgeschlagen, den Wert $g = (x, y) \in G$ so zu wählen, dass $x\theta_1(y) \in SL(2, \mathbb{Z}_p)$ Ordnung p hat. Da $\text{ord}_G\ g \mid \text{ord}_{SL(2,\mathbb{Z}_p)}(x\theta_1(y))$ hätten damit auch g und $Inn(g)$ die Ordnung p. Konkret wird empfohlen, $x\theta_1(y)$ von der Form $x\theta_1(y) = A(I + c\delta_{12})A^{-1}$ für ein $A \in SL(2, \mathbb{Z}_p)$ und ein $c \in \mathbb{Z}_p$ zu wählen. In Abschnitt 2.7.2 wurde gezeigt, dass das Diskreter Logarithmus Problem für Instanzen dieser Form einfach ist. Im selben Abschnitt wurde sogar gezeigt, dass das Diskreter Logarithmus Problem leicht ist in allen zyklischen Untergruppen von $SL(2, \mathbb{Z}_p)$ der Ordnung p. Das in [42] vorgeschlagene Konstruktionsprinzip ist also unsicher und sollte nicht verwendet werden. **Die in [42] vorgeschlagenen Parameter erlauben also eine Berechnung des geheimen Schlüssels a aus dem öffentlichen Schlüssel $\mathbf{Inn(g), Inn(g^a)}$.**

Dieser Angriff kann verhindert werden, indem man die Parameter folgendermaßen wählt: Seien p, q Primzahlen mit $q \mid p-1$. Dann hat $\mathbb{Z}_p^*$ eine eindeutig bestimmte Untergruppe G_q der Ordnung q. Wähle $a \in G_q$ mit $a \neq 1$ und $b \in \mathbb{Z}_p$. In Abschnitt 2.7.2 wurde gezeigt, dass dann die Matrix $\begin{pmatrix} a & b \\ 0 & a^{-1} \end{pmatrix}$ die Ordnung q hat und das DLP in der von ihr erzeugten Untergruppe von $SL(2, \mathbb{Z}_p)$ äquivalent zum DLP in G_q ist.
Wählt man $g = (x, y) \in G$ so, dass $x\theta_1(y)$ eine Matrix dieser Form ist, so ist es zwar möglich, aus den gegebenen Abbildungen $Inn(g)$ und $Inn(g^a)$ die Werte $\pm x\theta_1(y)$ und $\pm(x\theta_1(y))^a$ zu extrahieren. Unter der Annahme, dass das DLP schwer ist in G_q kann jedoch der geheime Schlüssel a nicht berechnet werden.

Das spezielle Konjugationsproblem in $\mathbf{SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p}$: Wir haben bereits gezeigt, dass durch Lösen des speziellen Konjugationsproblems in $SL(2, \mathbb{Z}_p)$ aus einem gegebenen inneren Automorphismus $Inn(g)$ mit $g = (x_1, y_1) \in SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ der Wert $\pm x_1\theta_1(y_1)$ extrahiert werden kann. Für jedes $y_2 \in \mathbb{Z}_p$ lässt sich durch $x_2 = \pm(x_1\theta_1(y_1)) \cdot \theta_1(-y_2)$ ein $x_2 \in SL(2, \mathbb{Z}_p)$ berechnen mit $x_1\theta_1(y_1) = \pm x_2\theta_1(y_2)$. Satz 18 auf der vorherigen Seite sagt aus, dass für $h = (x_2, y_2)$ dann $Inn(g) = Inn(h)$ gilt. Das beweist den folgenden Satz.

Satz 19 *Das spezielle Konjugationsproblem ist effizient lösbar in $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$.*

4.1 Effizienz von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$

In diesem Abschnitt soll untersucht werden, wie effizient MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ im Vergleich zu den etablierten Public-Key Verfahren wie RSA [45] oder den DL-basierten Verfahren ist. Da die Effizienz von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ eng mit der Effizienz von MOR auf $SL(2,\mathbb{Z}_p)$ verbunden ist und MOR auf der speziellen linearen Gruppe noch eine zentrale Rolle einnehmen wird, analysieren wir zunächst die Effizienz von MOR auf $SL(2,\mathbb{Z}_p)$.

Effizienz von MOR auf $\mathbf{SL(2,\mathbb{Z}_p)}$: Bei RSA und MOR auf $SL(2,\mathbb{Z}_p)$ würde man in der Praxis Moduln ähnlicher Größe verwenden. Aus diesem Grunde verwenden wir für einen Vergleich der Effizienz der beiden Verfahren, die Anzahl der Multiplikationen in $\mathbb{Z}_p$ im Falle von MOR bzw. in $\mathbb{Z}_n^*$ im Falle von RSA.
Zunächst betrachten wir den Aufwand, der nötig ist, um Potenzen in $< Inn(g) >$ zu bestimmen. Dabei gehen wir davon aus, dass $Inn(g)$ durch die Bilder $Inn(g)(S)$ und $Inn(g)(T)$ der Generatoren $S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ und $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ von $SL(2,\mathbb{Z}_p)$ gegeben ist. Um $Inn(g^2)$ zu berechnen, müssen also die beiden Funktionswerte $Inn(g^2)(S)$ und $Inn(g^2)(T)$ berechnet werden. Dazu müssen zunächst $Inn(g)(S)$ und $Inn(g)(T)$ als Produkte der Generatoren $Inn(g)(S) = T^{i_1}ST^{i_2}ST^{i_3}$ und $Inn(g)(T) = T^{j_1}ST^{j_2}ST^{j_3}$ dargestellt werden. Nach Abschnitt 2.7.1 sind dafür eine Inversion und zwei Multiplikationen in $\mathbb{Z}_p$ notwendig. Anschließend können

$$\begin{aligned} Inn(g^2)(S) &= Inn(g)(Inn(g)(S)) = Inn(g)(T^{i_1}ST^{i_2}ST^{i_3}) \\ &= (Inn(g)(T))^{i_1} Inn(g)(S)(Inn(g)(T))^{i_2} Inn(g)(S)(Inn(g)(T))^{i_3} \end{aligned}$$

und analog $Inn(g^2)(T)$ berechnet werden.

In Abschnitt 2.7.2 wurde gezeigt, dass für $g = \begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix} \in SL(2,\mathbb{Z}_p)$

$$\begin{aligned} (g(I + c\delta_{12})g^{-1})^i &= g(I + c\delta_{12})^i g^{-1} \\ &= \begin{pmatrix} 1 - cg_1g_3 & c(g_1)^2 \\ -c(g_3)^2 & 1 + cg_1g_3 \end{pmatrix} \end{aligned}$$

gilt. Da sich die Matrix T auch als $T = I + \delta_{12}$ darstellen lässt, ergibt sich

$$\begin{aligned} Inn(g)(T) &= \begin{pmatrix} 1 - g_1g_3 & (g_1)^2 \\ -(g_3)^2 & 1 + g_1g_3 \end{pmatrix} =: \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \\ &\text{und} \\ (Inn(g)(T))^i &= \begin{pmatrix} 1 - cg_1g_3 & c(g_1)^2 \\ -c(g_3)^2 & 1 + cg_1g_3 \end{pmatrix} \\ &= \begin{pmatrix} 1 + i(x_1 - 1) & ix_2 \\ ix_3 & 1 + i(x_4 - 1) \end{pmatrix}. \end{aligned}$$

Ist $Inn(g)(T)$ gegeben, so kann $(Inn(g)(T))^i$ also mit 4 Multiplikationen berechnet werden. Um $Inn(g^2)(T)$ bzw. $Inn(g^2)(S)$ zu berechnen, müssen dann noch jeweils 4 Matrixmultiplikationen, d.h. jeweils 32 Multiplikationen in $\mathbb{Z}_p$, ausgeführt werden. Insgesamt sind also 92 Multiplikationen in $\mathbb{Z}_p$ nötig, um $Inn(g^2)$ aus $Inn(g)$ zu berechnen.[11]
Für die Exponentiation mit einer n-bit Zahl, müssen ca. $\frac{3}{2}n$ Multiplikationen ausgeführt werden, in unserem Falle also etwa $\frac{3}{2}\log(\mathrm{ord}(g))$ Multiplikationen. Da während des Verschlüsselungsvorganges die Potenzen $Inn(g^b)$ und $Inn(g^{ab})$ von $Inn(g)$ bzw. $Inn(g^a)$ berechnet werden müssen, sind ungefähr $2 \cdot \frac{3}{2} \cdot 92 \cdot \log(\mathrm{ord}(g)) = 276 \cdot \log(\mathrm{ord}(g))$ Multiplikationen nötig. Hinzu kommen noch 46 Multiplikationen, um $Inn(g^{ab})(m)$ für eine gegebene Nachricht $m \in SL(2,\mathbb{Z}_p)$ zu berechnen.
Wählt man $g \in SL(2,\mathbb{Z}_p)$ so, dass $\mathrm{ord}(g) = q$ mit p, q prim und $q \mid p-1$, so sollten $\log p \geq 1024$ und $\log q \geq 160$ gewählt werden, um sicher gegen die bekannten Algorithmen zur Berechnung diskreter Logarithmen zu sein. In diesem Falle müssten also etwa $276 \cdot 160 + 46 = 44206$ Multiplikationen bei der Verschlüsselung und $138 \cdot 160 + 46 = 22126$ Multiplikationen bei der Entschlüsselung[12] berechnet werden.

Im Vergleich dazu müssen bei RSA mit einem 1024-Bit Modul ungefähr $\frac{3}{2} \cdot 1024 = 1536$ modulare Multiplikationen bei der Entschlüsselung und $\frac{3}{2} \cdot 32 = 48$ modulare Multiplikationen bei der Verschlüsselung ausgeführt werden. Dabei wurde vorausgesetzt, dass für den öffentlichen Exponenten ein kleiner Wert gewählt wurde.[13] Mit dieser Wahl der Parameter wäre die Verschlüsselung mit RSA also ca. 1000-mal und die Entschlüsselung ca. 20-mal schneller als bei MOR auf $SL(2,\mathbb{Z}_p)$.[14]
Ähnlich ungünstig fällt ein Vergleich mit den gängigen Algorithmen aus, die auf dem Diskreter Logarithmus Problem beruhen. Benutzt man für ElGamal ebenfalls eine Untergruppe G_q primer Ordnung von $\mathbb{Z}_p^*$ mit $|q| = 160$ und $|p| = 1024$, so müssen für Ver- und Entschlüsselung $2 \cdot \frac{3}{2} \cdot 160 = 480$ bzw. $\frac{3}{2} \cdot 160 = 240$ modulare Multiplikationen berechnet werden, was bedeutet, dass ElGamal in beiden Fällen ca. 90-mal schneller ist als MOR auf $SL(2,\mathbb{Z}_p)$.

[11] Zählt man die beiden Inversionen in $\mathbb{Z}_p^*$ noch mit, die in etwa so aufwändig sind, wie Multiplikationen in $\mathbb{Z}_p$, so kommt man sogar auf 94 Multiplikationen.

[12] Bei der Entschlüsselung muss nur eine Potenz, nämlich $Inn(g^{ab}) = (Inn(g^b))^a$, berechnet werden.

[13] Häufig gewählte Werte für den öffentlichen RSA Exponenten sind $e = 3$ und $e = 2^{16} + 1 = 65537$.

[14] RSA-Moduln der Länge 1024 Bits, sind nach dem heutigen Stand der Technik nicht mehr empfehlenswert (siehe [32]) Die Wahl von 1024 Bits als Größe der benutzten Primzahl p als auch als Größe des benutzten RSA-Moduls erlaubt einen Vergleich der Effizienz beider Verfahren (in beiden Fällen gemessen in modularen Multiplikationen mit einem 1024 Bit Modul). Selbst wenn man den RSA-Modul größer wählt und die benötigten Bitoperationen vergleicht, ändert sich das Verhältnis der benötigten Rechenzeiten nur unwesentlich.

Effizienz von MOR auf $\mathbf{SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p}$: Wir betrachten nun die Effizienz von MOR, wenn die Gruppe $G = SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ benutzt wird. Die Menge $\{(T,0),(S,0),(I,1)\}$ erzeugt die Gruppe $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$. Ist $a = T^{j_1}ST^{j_2}ST^{j_3} \in SL(2,\mathbb{Z}_p)$ eine Darstellung von $a \in SL(2,\mathbb{Z}_p)$, so ist $(a,b) = (T,0)^{j_1}(S,0)(T,0)^{j_2}(S,0)(T,0)^{j_3}(I,1)^b$ eine Darstellung des Elements $(a,b) \in SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$.

Wir setzen $(\bar{S},0) = Inn(g)((S,0))$, $(\bar{T},0) = Inn(g)((T,0))$ und $(R,1) = Inn(g)((I,1))$. Seien $\bar{S} = T^{i_1}ST^{i_2}ST^{i_3}$ und $R = T^{j_1}ST^{j_2}ST^{j_3}$ Darstellungen von $\bar{S}, R \in SL(2,\mathbb{Z}_p)$. Dann gilt

$$\begin{aligned} Inn(g^2)((S,0)) &= Inn(g)(Inn(g)((S,0))) = Inn(g)((\bar{S},0)) \\ &= Inn(g)((T,0)^{i_1}(S,0)(T,0)^{i_2}(S,0)(T,0)^{i_3}) \\ &= (\bar{T},0)^{i_1}(\bar{S},0)(\bar{T},0)^{i_2}(\bar{S},0)(\bar{T},0)^{i_3} \\ &= (\bar{T}^{i_1}\bar{S}\bar{T}^{i_2}\bar{S}\bar{T}^{i_3},0). \end{aligned}$$

Das Berechnen von $Inn(g^2)((S,0))$ bei gegebenem $Inn(g)$ in $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ ist also genauso aufwändig wie das Berechnen von $Inn(h^2)(S)$ bei gegebenem $Inn(h)$ in $SL(2,\mathbb{Z}_p)$. Die Berechnung von $Inn(g^2)((T,0))$ verläuft vollkommen analog. Zur Berechnung von $Inn(g^2)((I,1))$ muss eine zusätzliche Matrixmultiplikation ausgeführt werden:

$$\begin{aligned} Inn(g^2)((I,1)) &= Inn(g)(Inn(g)((I,1))) \;=\; Inn(g)((R,1)) \\ &= Inn(g)((T,0)^{j_1}(S,0)(T,0)^{j_2}(S,0)(T,0)^{j_3}(I,1)) \\ &= (\bar{T}^{i_1}\bar{S}\bar{T}^{i_2}\bar{S}\bar{T}^{i_3},0)\cdot(R,1) \\ &= (\bar{T}^{i_1}\bar{S}\bar{T}^{i_2}\bar{S}\bar{T}^{i_3}R,1). \end{aligned}$$

Da in unserem Fall $\mathrm{ord}(g) = q$ gewählt wurde, sind für die Berechnung von Potenzen von $Inn(g)$ etwa $\frac{3}{2}\cdot 146 \cdot \log q = 219 \cdot \log q$ Multiplikationen in $\mathbb{Z}_p$ nötig.

Während der Ver- und Entschlüsselung ist außer den Potenzen von $Inn(g)$ zusätzlich jeweils ein Funktionswert zu berechnen. Seien $Inn(g)$ und $(m_1,m_2) \in G$ gegeben. Dann ist

$$Inn(g)((m_1,m_2)) = Inn(g)((m_1,0)) \cdot Inn(g)((I,m_2)).$$

Zur Berechnung von $Inn(g)((m_1,0))$ sind 46 Multiplikationen in $\mathbb{Z}_p$ nötig. Der Wert $Inn(g)((I,m_2))$ ergibt sich für $(R,1) = Inn(g)((I,1))$ als

$$\begin{aligned} Inn(g)((I,m_2)) &= Inn(g)((I,1)^{m_2}) \\ &= (Inn(g)((I,1)))^{m_2} \\ &= (R,1)^{m_2}. \end{aligned}$$

Zur Berechnung von $Inn(g)((I,m_2))$ sind also noch einmal etwa $\frac{3}{2}\log p$ Multiplikationen in $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ nötig. Um das Produkt $(g_1\theta_1(h_1)g_2\theta_1(-h_1), h_1 + h_2)$ zweier Werte

$(g_1, h_1), (g_2, h_2) \in SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ zu berechnen, müssen der Funktionswert $\theta_1(h_1)$ sowie 3 Multiplikationen in $SL(2, \mathbb{Z}_p)$ berechnet werden.
Wir nehmen an, dass zur Berechnung von $(R, 1)^{m_2}$ ein Square-and-Multiply Algorithmus verwendet wird, d.h. zunächst werden die Zweierpotenzen $(R, 1)^{2^k}$ berechnet (Square-Schritt) und dann entsprechend der Binärdarstellung von m_2 multipliziert (Multiply-Schritt). Zur Berechnung von $(R, 1)^{2^k} = (R, 1)^{2^{k-1}} \cdot (R, 1)^{2^{k-1}}$ wird der Wert $\theta_1(2^{k-1})$ benötigt. Da zur Berechnung von $(R, 1)^{2^{k-1}}$ jedoch schon $\theta_1(2^{k-2})$ bestimmt werden musste, kann $\theta_1(2^{k-1})$ unter Ausnutzung der Homomorphie-Eigenschaft von θ_1 mit nur einer Matrixmultiplikation berechnet werden. Die Funktionswerte von θ_1, die im Multiply-Schritt benötigt werden, wurden bereits im Square-Schritt berechnet und können zwischengespeichert werden. Wir schätzen den Aufwand zur Berechnung eines Funktionswerts von θ_1 in diesem speziellen Fall mit dem Aufwand für eine Matrixmultiplikation in $SL(2, \mathbb{Z}_p)$ ab.
In unserem Fall schlägt eine Multiplikation in $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ demnach mit 4 Matrixmultiplikationen, d.h. 32 Multiplikationen in $\mathbb{Z}_p$ und die Berechnung von $Inn(g)((I, m_2))$ insgesamt mit $32 \cdot \frac{3}{2} \log p = 48 \log p$ Multiplikationen in $\mathbb{Z}_p$ zu Buche. Hinzu kommen 8 Multiplikationen, um die erhaltenen Ergebnisse $Inn(g)((m_1, 0))$ und $Inn(g)((I, m_2))$ zu multiplizieren. Insgesamt sind für die Berechnung des Funktionswertes also $54 + 48 \cdot \log p$ Multiplikationen in $\mathbb{Z}_p$ nötig.

Addiert man die erhaltenen Werte, so ergibt sich bei MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ für die Verschlüsselung ein Aufwand von $2 \cdot 219 \cdot \log q + 54 + 48 \cdot \log p = 438 \cdot \log q + 54 + 48 \cdot \log p$ und für die Entschlüsselung ein Aufwand von $219 \cdot \log q + 54 + 48 \cdot \log p$ Multiplikationen in $\mathbb{Z}_p$. Für $\log p = 1024$ und $\log q = 160$ ergeben sich 119286 bzw. 49152 Multiplikationen in $\mathbb{Z}_p$. MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ ist damit bei der Verschlüsselung ca. 2500-mal und bei der Entschlüsselung ca. 32-mal langsamer als RSA.

4.2 Die Betriebsmodi von MOR

Um die Effizienz des MOR Schemas zu erhöhen, schlagen die Autoren in [42] vor, den Exponenten b, der in der Grundversion des MOR Schemas für jede Verschlüsselung neu gewählt wird, für mehrere Verschlüsselungen fest zu lassen. Der Absender müsste den Exponenten b dann nur einmal wählen und nur einmal die sehr aufwändigen Berechnungen von $Inn(g^b)$ und $Inn(g^{ab})$ machen. Bei den folgenden Kommunikationen mit dem selben Empfänger könnten diese Berechnungen dann entfallen.

Das resultierende Verfahren ist deterministisch, d.h. gleiche Klartexte werden auch immer auf die gleichen Geheimtexte abgebildet. Dies kann durch Benutzung des ebenfalls in [42] vorgeschlagenen probabilistischen Padding-Verfahrens verhindert werden:
Sei $M \in \mathbb{Z}_p$, $M \neq 0$, die zu verschlüsselnde Nachricht. Wähle $r_1, r_2 \in_R \mathbb{Z}_p$. Die Matrix $m = \begin{pmatrix} M & r_1 \\ r_2 & \frac{1+r_1 r_2}{M} \end{pmatrix} \in SL(2, \mathbb{Z}_p)$ wird anschließend mit MOR auf $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ verschlüsselt.

In $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ ist die zweite Komponente invariant unter Konjugation, d.h. für $(a, b), (x, y) \in G$ ist $(x, y)(a, b)(x, a)^{-1} = (\bar{a}, b)$ für ein $\bar{a} \in SL(2, \mathbb{Z}_p)$. Deshalb darf die zu verschlüsselnde Nachricht ausschließlich in die erste Komponente kodiert werden (siehe Kapitel 4). Ein weiterer Vorschlage zur Effizienzsteigerung ist deshalb, die zweite Komponente b des zu konjugierenden Gruppenelements $(a, b) \in G$ auf Null zu setzen, d.h. die inneren Automorphismen auf den Normalteiler $SL(2, \mathbb{Z}_p) \times \{0\}$ zu beschränken.

Insgesamt ergeben sich die folgenden 4 Varianten von MOR auf $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$, die jeweils mit und ohne Padding benutzt werden können:

1. Grundversion von MOR

   Effizienz (in Multiplikationen in $\mathbb{Z}_p$):
   Verschlüsselung: $54 + 438 \cdot \log q + 48 \cdot \log p$
   Entschlüsselung: $54 + 219 \cdot \log q + 48 \cdot \log p$

2. MOR mit festem Exponenten b

   Effizienz (in Multiplikationen in $\mathbb{Z}_p$):
   Verschlüsselung: $54 + 48 \cdot \log p$
   Entschlüsselung: $54 + 48 \cdot \log p$

   Hinzu kommen einmalig $438 \cdot \log q$ Multiplikationen auf Senderseite und $219 \cdot \log q$ auf Empfängerseite.

3. MOR mit Einschränkung auf $SL(2, \mathbb{Z}_p) \times \{0\}$

   Seien $g = (x, y), (a, 0) \in G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$. Dann gilt

$$\begin{aligned} Inn(g)((a,0)) &= (x,y)(a,0)(x,y)^{-1} \\ &= ((x\theta_1(y))a(x\theta_1(y))^{-1}, 0) \end{aligned}$$

   und

$$\begin{aligned} Inn(g^n)((a,0)) &= (x,y)^n(a,0)(x,y)^{-n} \\ &= ((x\theta_1(y))^n a(x\theta_1(y))^{-n}, 0). \end{aligned}$$

   MOR mit Einschränkung auf $SL(2, \mathbb{Z}_p) \times \{0\}$ ist äquivalent zu MOR auf $SL(2, \mathbb{Z}_p)$: Die Automorphismen $Inn(g)$ und $Inn(g^a)$ in $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ entsprechen den Automorphismen $Inn(x\theta_1(y))$ bzw. $Inn((x\theta_1(y))^a)$ in $SL(2, \mathbb{Z}_p)$. Der einfacheren Notation wegen werden wir in folgenden Abschnitten MOR auf $SL(2, \mathbb{Z}_p)$ genauer untersuchen. Die Ergebnisse gelten analog für MOR mit Einschränkung auf $SL(2, \mathbb{Z}_p) \times \{0\}$.

   Effizienz (in Multiplikationen in $\mathbb{Z}_p$):
   Verschlüsselung: $46 + 276 \cdot \log p$
   Entschlüsselung: $46 + 138 \cdot \log p$

4. MOR mit Einschränkung auf $SL(2, \mathbb{Z}_p) \times \{0\}$ und festem Exponenten b

   Effizienz (in Multiplikationen in $\mathbb{Z}_p$):
   Verschlüsselung: 46
   Entschlüsselung: 46

   Hinzu kommen einmalig $276 \cdot \log p$ Multiplikationen auf Senderseite und $138 \cdot \log p$ auf Empfängerseite.

4.3 Angriffe auf MOR auf $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$

Wir betrachten zunächst MOR auf $SL(2, \mathbb{Z}_p)$ und zeigen im folgenden Abschnitt, wie sich die vorgestellten Angriffe auf MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ übertragen lassen. Die angegebenen Verfahren funktionieren auch, wenn die Gruppe $GL(2, \mathbb{Z}_p)$ anstatt $SL(2, \mathbb{Z}_p)$ verwendet wird. Der größeren Allgemeinheit wegen formulieren wir unsere Ergebnisse deshalb für $GL(2, \mathbb{Z}_p)$.

Die vorgestellten Angriffe machen sich im Wesentlichen zu Nutze, dass bei der Konjugation in $GL(2, \mathbb{Z}_p)$ die Determinante, die Spur und die Ordnung invariant sind.

Satz 20 *Seien $X, A \in GL(2, \mathbb{Z}_p)$ und $Y = XAX^{-1}$ Dann gilt:*

1. *det (A) = det (Y)*
2. *Spur (A) = Spur (Y)*
3. *Für $A \notin Z(X)$ und $\mathrm{ord}(A) = q$ mit q prim gilt ord (A) = ord (Y).*

Beweis: Aussage 1 ist eine Folgerung aus dem Determinantenmultiplikationssatz, da det $(X^{-1}) = (\det(X))^{-1}$. Der Beweis von Aussage 3 verläuft analog zum Beweis von Lemma 3 auf Seite 46. Bleibt Aussage 2 zu zeigen.
Seien $A = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$, $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$ und damit $Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}$ aus $GL(2, \mathbb{Z}_p)$.
Da $X^{-1} = \begin{pmatrix} \frac{x_4}{\det(X)} & \frac{-x_2}{\det(X)} \\ \frac{-x_3}{\det(X)} & \frac{x_1}{\det(X)} \end{pmatrix}$ folgt $y_1 = \frac{1}{\det(X)}(a_1x_1x_4 + a_3x_2x_4 - a_2x_1x_3 - a_4x_2x_3)$ und $y_4 = \frac{1}{\det(X)}(a_2x_3x_1 + a_4x_4x_1 - a_1x_3x_2 - a_3x_4x_2)$ und damit Spur $(Y) = y_1 + y_4 = \frac{1}{\det(X)}((a_1 + a_4)(x_1x_4 - x_2x_3)) = \frac{1}{\det(X)}((a_1 + a_4)\det(X)) = a_1 + a_4 = \text{Spur } (A)$. □

4.3.1 Ciphertext-Only Angriff auf die Grundversion von MOR

In diesem Abschnitt untersuchen wir die Anfälligkeit von MOR auf $GL(2, \mathbb{Z}_p)$ gegenüber Ciphertext-Only Angriffen. In einem Ciphertext-Only Angriff stehen einem Angreifer für seine Analyse lediglich die öffentlichen Parameter des Systems, der öffentliche Schlüssel eines Teilnehmers sowie eine oder mehrere mit diesem Schlüssel erstellte Geheimtexte zur Verfügung. Die dazugehörigen Klartexte sind ihm nicht bekannt. Ciphertext-Only Angriffe gehören zu den schwächsten Angriffen, die ein Angreifer ausführen kann. Wir werden zeigen, dass bei Benutzung von MOR auf $GL(2, \mathbb{Z}_p)$ allein aus Kenntnis eines Geheimtextes Informationen über den darin verschlüsselten Klartext berechnet werden können.

Sei $g \in GL(2, \mathbb{Z}_p)$ und mit $C = Inn(g^{ab})(M) = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$ ein MOR Ciphertext von $M = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \in GL(2, \mathbb{Z}_p)$ gegeben.

Der Geheimtext C liefert bereits Informationen über den verschlüsselten Klartext M. Durch die Invarianz von Spur und Determinante erhält man

- $m_1 + m_4 = c_1 + c_4$ und
- $m_1 m_4 - m_2 m_3 = c_1 c_4 - c_2 c_3$.

Allein durch diese beiden Gleichungen lässt sich bereits sagen, dass der gesucht Klartext M von der Form

$$M = \begin{pmatrix} c_1 + c_4 - m_4 & \frac{(c_1+c_4-m_4)m_4 - \det(C)}{m_3} \\ m_3 & m_4 \end{pmatrix}$$

sein muss.

Extraktion eines $\mathbf{h \in Z(g)}$ aus $\mathbf{Inn(g)}$: Bisher haben wir alle Informationen über den verschlüsselten Klartext $M \in GL(2,\mathbb{Z}_p)$ aus $C = Inn(g^{ab})(M)$ extrahiert. Neben C sind aber auch die Abbildungen $Inn(g)$ und $Inn(g^a)$, die Teil des öffentlichen Schlüssels sind, und die Abbildung $Inn(g^b)$, die Teil des Geheimtextes ist, bekannt. Dieses Wissen verwenden wir jetzt dazu, ein Element h des Zentralisators $Z(g)$ von g zu berechnen.

Unabhängig davon, wie die Abbildung $Inn(g)$ gegeben ist, lässt sich aus Kenntnis von $Inn(g)$ ein Element $h \in Z(g)$ aus dem Zentralisator von g berechnen: Da das spezielle Konjugationsproblem leicht ist in $GL(2,\mathbb{Z}_p)$, kann ein Element $h \in GL(2,\mathbb{Z}_p)$ berechnet werden mit $Inn(g) = Inn(h)$ (siehe Abschnitt 2.7.4 auf Seite 29). Dieses Element hat nach Satz 4 auf Seite 23 die Form $h = g \cdot z$ mit $z \in Z(GL(2,\mathbb{Z}_p))$ und damit gilt $h \in Z(g)$.

Verbesserung des Angriffes: Sei $h = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix} \in Z(g)$. Dann kann aus dem gegebenen Geheimtext $C = Inn(g^{ab})(M) = g^{ab} M g^{-ab}$ ein Geheimtext $\widehat{C}$ von $h \cdot M$ berechnet werden: $\widehat{C} := h \cdot C = h \cdot g^{ab} M g^{-ab} = g^{ab}(h \cdot M) g^{-ab}$. Nutzt man erneut die Invarianz der Spur unter Konjugation aus, so ergibt sich für $\widehat{C} = \begin{pmatrix} \widehat{c}_1 & \widehat{c}_2 \\ \widehat{c}_3 & \widehat{c}_4 \end{pmatrix}$, dass die Gleichung $\widehat{c}_1 + \widehat{c}_4 = \text{Spur}(\widehat{C}) = \text{Spur}(h \cdot M) = h_1 m_1 + h_2 m_3 + h_3 m_2 + h_4 m_4$ erfüllt ist. Insgesamt hat man also folgende Informationen über den gesuchten Klartext M:

- $m_1 + m_4 = c_1 + c_4$,
- $m_1 m_4 - m_2 m_3 = c_1 c_4 - c_2 c_3$ und
- $h_1 m_1 + h_2 m_3 + h_3 m_2 + h_4 m_4 = \widehat{c}_1 + \widehat{c}_4$.

Der gesuchte Klartext M ist für $m_4 \in \mathbb{Z}_p$ also von der Form

$$M = \begin{pmatrix} c_1 + c_4 - m_4 & \frac{\widehat{c}_1 + \widehat{c}_4 - h_1(c_1+c_4) - h_2 m_3 - (h_4 - m_4)m_4}{h_3} \\ \frac{((c_1+c_4-m_4)m_4 - c_1 c_4 + c_2 c_3)h_3}{\widehat{c}_1 + \widehat{c}_4 - h_1(c_1+c_4) - h_2 m_3 - (h_4-m_4)m_4} & m_4 \end{pmatrix}.$$

Die Menge der möglichen Klartexte für Geheimtext C konnte damit auf einen (im Sicherheitsparameter) exponentiell kleinen Anteil des Klartextraumes reduziert werden. Insbesondere kann der komplette Klartext aus dem Geheimtext rekonstruiert werden, wenn eine Komponente von ihm bekannt ist.

Die Grundversion des MOR Schema auf $SL(2, \mathbb{Z}_p)$ sollte auf keine Fall ohne Padding verwendet werden. Verwendet man eine Padding-Technik, bei dem die zu verschlüsselnde Nachricht in eine Komponente der Matrix M kodiert wird, so liefert dieser Angriff keine Informationen über den verschlüsselten Klartext.

4.3.2 Known-Plaintext Angriff auf MOR mit festen Exponenten

Die im letzten Abschnitt demonstrierten Angriffe stellen sich als besonders stark heraus, wenn MOR mit einem festen Verschlüsselungsexponenten verwendet wird. Dazu nehmen wir an, dass der Angreifer einen Known-Plaintext Angriff ausführt, d.h. er verfügt über Paare bestehend aus Klar- und zugehörigem Geheimtext. Known-Plaintext Angriffe sind zwar stärkere Angriffe als die im letzten Abschnitt benutzten Ciphertext-Only Angriffe, da dem Angreifer zusätzlich zum Geheimtext noch der dazugehörige Klartext zur Analyse zur Verfügung steht. Bei ihnen handelt es sich aber immer noch um sehr schwache Angriffe.

Wir betrachten zunächst die Variante, in der MOR auf $GL(2, \mathbb{Z}_p)$ mit festem Verschlüsselungsexponenten und ohne Padding verwendet wird. In diesem Fall ergibt sich der zu einer Nachricht $M \in GL(2, \mathbb{Z}_p)$ gehörende Geheimtext C als $C = Inn(g^{ab})(M) = g^{ab} \cdot M \cdot g^{-ab}$. Zusätzlich zu dem im letzten Abschnitt beschriebenen Ciphertext-Only Angriff ist diese Variante extrem anfällig gegen Known-Plaintext Angriffe. Sind nämlich Paare aus Klar- und zugehörigem Geheimtext bekannt, so stellen diese Paare Instanzen des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ dar, die genutzt werden können, um das spezielle Konjugationsproblem für $Inn(g^{ab})$ zu lösen. Wie in Abschnitt 2.7.4 gezeigt wurde, reichen zwei geeignete Instanzen aus, um das spezielle Konjugationsproblem in $GL(2, \mathbb{Z}_p)$ zu lösen. Sind also zwei Paare $(M_1, C(M_1))$ und $(M_2, C(M_2))$ bestehend aus den Klartexten M_2 und M_2 und den zugehörigen Geheimtexten $C(M_1)$ und $C(M_2)$ mit $M_1 \notin Z(M_2)$ bekannt, so kann ein Element $\hat{g} \in GL(2, \mathbb{Z}_p)$ mit $Inn(\hat{g}) = Inn(g^{ab})$ berechnet. Da der Exponent b fest gewählt wurde, werden auch die folgenden Klartexte mit der selben Funktion $Inn(\hat{g}) = Inn(g^{ab})$ verschlüsselt und können effizient entschlüsselt werden.

An dieser Schwäche ändert auch die Benutzung des vorgeschlagenen Padding-Verfahrens nichts. Sei $m \in \mathbb{Z}_p$ die zu verschlüsselnde Nachricht. Der zugehörige Geheimtext hat dann die Form

$$C(m) = Inn(g^{ab})(M) = g^{ab} \cdot \begin{pmatrix} m & r_1 \\ r_2 & r_3 \end{pmatrix} \cdot g^{-ab},$$

wobei die Komponenten $r_1, r_2, r_3 \in_R \mathbb{Z}_p$ zufällig mit $\det(M) \neq 0$ gewählt werden. Hier wird eine leicht abgewandelte Version des Padding-Verfahrens für die Gruppe $GL(2, \mathbb{Z}_p)$

benutzt. Fordert man zusätzlich, dass $r_3 = \frac{1+r_1 r_2}{m}$ gilt, so erhält man das in Abschnitt 4.2 auf Seite 53 definierte Padding-Verfahren für $SL(2, \mathbb{Z}_p)$ als Spezialfall. Da wegen der Invarianz der Determinante unter Konjugation der Wert $\det(M)$ bekannt ist, bringt eine konkrete Wahl von $\det(M)$ keine Vorteile. Aus diesem Grund betrachten wir den allgemeinen Fall mit $M \in GL(2, \mathbb{Z}_p)$.

Ist die Nachricht $m \in \mathbb{Z}_p$ sowie der zugehörige Geheimtext $C(M)$ bekannt, so kann ein Angreifer den Ciphertext-Only Angriff aus dem letzten Abschnitt benutzen, um zunächst die Matrix $M \in GL(2, \mathbb{Z}_p)$ komplett zu berechnen. Er ist dann in der selben Situation wie bei MOR auf $GL(2, \mathbb{Z}_p)$ mit festen Exponenten ohne Padding und kann alle zukünftig versandten Geheimtexte effizient entschlüsseln.

Wie schwach MOR auf $GL(2, \mathbb{Z}_p)$ mit festem Exponenten und dem vorgeschlagenen Padding ist, zeigt der folgende Angriff, bei dem wir annehmen, dass der Angreifer nur einen Teil des Geheimtextes besitzt und den öffentlichen Schlüssel des Empfängers nicht kennt. Da wir MOR mit festem Verschlüsselungsexponenten b betrachten, nehmen wir an, dass die Funktion $Inn(g^b)$ als Teil des Geheimtextes nur bei der ersten Benutzung des Exponenten übertragen und fortan als bekannt vorausgesetzt wird. In diesem Fall besteht der übertragene Geheimtext also lediglich aus $C(m) = Inn(g^{ab})(M)$. Wir nehmen ferner an, dass der Angreifer die Übertragung von $Inn(g^b)$ nicht abfangen konnte und nicht im Besitz des öffentlichen Schlüssels des Empfängers ist. Er ist damit insbesondere auch nicht in der Lage aus $Inn(g)$ durch Lösen des speziellen Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ ein Element $h \in Z(g)$ zu berechnen. Wir werden zeigen, dass selbst in diesem Fall zwei Paare bestehend aus Klartext m und zugehörigem Geheimtext $C(m)$ ausreichen, um die Funktion $Inn(g^{ab})$ zu berechnen.

Seien mit $m_1, m_2 \in \mathbb{Z}_p$ zwei Nachrichten und die zugehörigen Geheimtexte

$$Inn(g^{ab})(M_1) = g^{ab} \cdot \begin{pmatrix} m_1 & r_1 \\ r_2 & r_3 \end{pmatrix} \cdot g^{-ab} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} = C$$

und

$$Inn(g^{ab})(M_2) = g^{ab} \cdot \begin{pmatrix} m_2 & s_1 \\ s_2 & s_3 \end{pmatrix} \cdot g^{-ab} = \begin{pmatrix} d_1 & d_2 \\ d_3 & d_4 \end{pmatrix} = D$$

gegeben. Über die Spuren kann der Angreifer $\mathrm{Spur}(C) = \mathrm{Spur}(M_1) = m_1 + r_3$ und $\mathrm{Spur}(D) = \mathrm{Spur}(M_2) = m_2 + s_3$ berechnen. Da ferner die Klartexte m_1 und m_2 bekannt sind, können damit insbesondere r_3 und s_3 berechnet werden. Über die Determinante $\det(C) = \det(M_1) = m_1 \cdot r_3 - r_1 \cdot r_2$ kann der Angreifer daraus $r_1 \cdot r_2$ und analog berechnen (und den Wert $s_1 \cdot s_2$ analog über die Determinante von D).

Nutzt man die Homomorphie-Eigenschaft der inneren Automorphismen aus, so erhält man $\mathrm{Spur}(C \cdot D) = \mathrm{Spur}(M_1 \cdot M_2) = m_1 m_2 + r_1 s_2 + r_2 s_1 + r_3 s_3$. Mittels der bereits bekannten Werte kann $r_1 s_2 + r_2 s_1$ berechnet werden.

Lösen der quadratischen Gleichung

$$x^2 - (r_1 s_2 + r_2 s_1)x + s_1 s_2 r_1 r_2 = 0$$

liefert die Werte $r_1 \cdot s_2$ und $r_2 \cdot s_1$ als Lösungen (hier muss geraten werden, welche der beiden Lösungen dem Wert $r_1 \cdot s_2$ bzw. dem Wert $r_2 \cdot s_1$ entspricht).

Das simultane Lösen der Instanzen M_1, C und M_2, D des CP in $GL(2,\mathbb{Z}_p)$ führt zu folgendem Gleichungssystem (dabei seien g_1, g_2, g_3, g_4 die Komponenten von g^{ab}):

$$\begin{array}{ccccccc} g_1 & + & & & \frac{s_2(c_4-m_1)-r_2(d_4-m_2)}{c_3(d_4-m_2)-d_3(c_4-m_1)} \cdot g_4 & = 0 \\ & g_2 & + & & (\frac{c_4-r_3}{c_3} + \frac{d_3 r_1 r_2 - c_3 r_1 s_2}{c_3(c_3(d_4-m_2)-d_3(c_4-m_1))}) \cdot g_4 & = 0 \\ & & g_3 & + & \frac{d_3 r_2 - c_3 s_2}{c_3(d_4-m_2)-d_3(c_4-m_1)} \cdot g_4 & = 0 \end{array}$$

Da $r_1 \cdot r_2$ und $r_1 \cdot s_2$ bekannt sind, kann s_2 als $s_2 = k \cdot r_2$ für ein $k \in \mathbb{Z}_p$ ausgedrückt werden. Man erhält $g_1 = u_1 r_2 g_4$, $g_2 = u_2 g_4$ and $g_3 = u_3 r_2 g_4$, wobei $u_1 = \frac{(d_4-m_2)-k(c_4-m_1)}{c_3(d_4-m_2)-d_3(c_4-m_1)}$, $u_2 = \frac{r_3-c_4}{c_3} + \frac{c_3 r_1 s_2 - d_3 r_1 r_2}{c_3(c_3(d_4-m_2)-d_3(c_4-m_1))}$ und $u_3 = \frac{k c_3 + d_3}{c_3(d_4-m_2)-d_3(d_4-m_1)}$, als Lösungsmenge des linearen Gleichungssystems.

Über die Gleichung $C = Inn(g^{ab})(M_1)$ ergibt sich

$$\begin{aligned} c_1 &= \frac{1}{\det(g^{ab})}(m_1 g_1 g_4 + r_2 g_2 g_4 - r_1 g_1 g_3 - r_3 g_2 g_3) \\ &= \frac{1}{\det(g^{ab})} r_2 g_4^2 (m_1 u_1 + u_2 - r_1 r_2 u_1 u_3 - r_3 u_2 u_3). \end{aligned}$$

Wir definieren $u_4 := \frac{r_2 g_4^2}{\det(g^{ab})}$.

Die Werte u_1, u_2, u_3 und u_4 reichen aus, um alle Geheimtexte, die unter Verwendung des selben Exponenten b verschlüsselt wurden, zu entschlüsseln:

Sei der Geheimtext

$$Inn(g^{ab})(M_3) = g^{ab} \cdot \begin{pmatrix} m_3 & t_1 \\ t_2 & t_3 \end{pmatrix} \cdot g^{-ab} = \begin{pmatrix} e_1 & e_2 \\ e_3 & e_4 \end{pmatrix} = E$$

gegeben. Der Klartext m_3 ergibt sich dann als

$$\begin{aligned} m_3 &= \frac{1}{\det(g^{ab})}(e_1 g_1 g_4 - e_3 g_1 g_2 + e_2 g_3 g_4 - e_4 g_1 g_2) \\ &= \frac{1}{\det(g^{ab})} r_2 (g_4)^2 (e_1 u_1 - e_3 u_1 u_2 + e_2 u_3 - e_4 u_2 u_3) \\ &= u_4(e_1 u_1 - e_3 u_1 u_2 + e_2 u_3 - e_4 u_2 u_3). \end{aligned}$$

In analoger Weise können auch die übrigen Komponenten der Matrix M_3 berechnet werden.

4.4 Anwendung der Angriffe auf MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$

Zu Beginn dieses Kapitels wurde bereits die Verwandtschaft von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ mit MOR auf $SL(2,\mathbb{Z}_p)$ gezeigt. Diese Verwandtschaft werden wir in diesem Abschnitt ausnutzen, um die im letzten Abschnitt für MOR auf $SL(2,\mathbb{Z}_p)$ vorgestellten Angriffe auf MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ zu übertragen. Aufgrund der größeren Allgemeinheit lassen wir wieder Matrizen aus $GL(2,\mathbb{Z}_p)$ zu, so dass faktisch die Gruppe $G = GL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ verwendet wird.

Seien $Inn(g)$ und $Inn(g^a)$ für $g \in G$ ein öffentlicher Schlüssel von MOR auf $GL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ und $M \in GL(2,\mathbb{Z}_p)$ die zu verschlüsselnde Nachricht. Der resultierende Geheimtext ist dann von der Form

$$\begin{aligned} Inn(g^{ab})((M,m)) &= (x,y)^{ab} \cdot (M,m) \cdot (x,y)^{-ab} \\ &= ((x\theta_1(y))^{ab} \cdot M\theta_1(m) \cdot (x\theta_1(y))^{-ab}\theta_1(-m), m). \end{aligned}$$

Wie im letzten Abschnitt betrachten wir neben der Grundversion von MOR wieder den Betriebsmodus, in dem der Exponent b für mehrere Verschlüsselungen fest gehalten wird. Beide Varianten betrachten wir wieder mit und ohne Benutzung des probabilistischen Padding-Verfahrens.

MOR auf $\mathbf{SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p}$ mit festem Exponenten ohne Padding: Wir betrachten zunächst die Version ohne Padding. Sei $M \in GL(2,\mathbb{Z}_p)$ die zu verschlüsselnde Nachricht. Der resultierende Geheimtext ist dann von der Form

$$\begin{aligned} Inn(g^{ab})((M,m)) &= (x,y)^{ab} \cdot (M,m) \cdot (x,y)^{-ab} \\ &= ((x\theta_1(y))^{ab} \cdot M\theta_1(m) \cdot (x\theta_1(y))^{-ab}\theta_1(-m), m). \end{aligned}$$

Wir betrachten wieder Known-Plaintext Angriffe, d.h. wir nehmen an, dass ein Angreifer über Paare von Klar- und zugehörigem Geheimtext verfügt. Aus der Nachricht M und der zweiten Komponente m des Geheimtextes kann der Wert $M\theta_1(m)$ berechnet werden. Da $\theta_1(m)$ bekannt ist, kann aus der ersten Komponente des Geheimtextes ferner $(x\theta_1(y))^{ab} \cdot M\theta_1(m) \cdot (x\theta_1(y))^{-ab}$ berechnet werden. Diese beiden Werte bilden eine Instanz des CP in $GL(2,\mathbb{Z}_p)$. Kennt man zwei Paare $(M_1, C(M_1))$ und $(M_2, C(M_2))$ von Klar- und zugehörigem Geheimtext, so können für $C(M_1) = (c_1, c_2)$ und $C(M_2) = (\hat{c_1}, \hat{c_2})$ mit $M_1\theta_1(c_2), (x\theta_1(y))^{ab} \cdot M_1\theta_1(c_2) \cdot (x\theta_1(y))^{-ab}$ und $M_2\theta_1(\hat{c_2}), (x\theta_1(y))^{ab} \cdot M_2\theta_1(\hat{c_2}) \cdot (x\theta_1(y))^{-ab}$ zwei Instanzen des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$ extrahiert werden. Ist die Bedingung $M_1 \cdot \theta_1(c_2) \notin Z(M_1 \cdot \theta_1(\hat{c_2}))$ erfüllt, so reichen diese bereits aus, um das SCP in $GL(2,\mathbb{Z}_p)$ zu lösen, d.h. ein Element $\hat{g} \in GL(2,\mathbb{Z}_p)$ mit $Inn(\hat{g}) = Inn((x\theta_1(y))^{ab})$ zu berechnen. Damit können alle folgenden Geheimtexte effizient entschlüsselt werden. Sei der Geheimtext zu einer unbekannten Nachricht $M_3 \in GL(2,\mathbb{Z}_p)$ gegeben durch

$$\begin{aligned} C(M_3) &= (\tilde{c_1}, \tilde{c_2}) \\ &= ((x\theta_1(y))^{ab} \cdot M_3\theta_1(\tilde{c_2}) \cdot (x\theta_1(y))^{-ab}\theta_1(-\tilde{c_2}), \tilde{c_2}). \end{aligned}$$

Die verschlüsselte Nachricht M_3 ergibt sich dann als

$$\begin{aligned}(Inn(\hat{g}^{-1})(\tilde{c_1}\theta_1(\tilde{c_2})))\theta_1(-\tilde{c_2}) &= (Inn(\hat{g}^{-1})((x\theta_1(y))^{ab}M_3\theta_1(\tilde{c_2})(x\theta_1(y))^{-ab}))\theta_1(-\tilde{c_2}) \\ &= M_3\theta_1(\tilde{c_2})\theta_1(-\tilde{c_2}) \\ &= M_3.\end{aligned}$$

MOR auf $\mathbf{SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p}$ mit festem Exponenten mit Padding: Genau wie bei MOR auf $GL(2,\mathbb{Z}_p)$ reichen auch hier trotz Benutzung des vorgeschlagenen Paddings zwei geeignete Paare bestehend aus Klar- und zugehörigem Geheimtext aus, um alle weiteren Geheimtexte effizient entschlüsseln zu können. Wir werden zeigen, dass bei Kenntnis eines Geheimtextes

$$C(m) = ((x\theta_1(y))^{ab} \cdot M\theta_1(c_2) \cdot (x\theta_1(y))^{-ab}\theta_1(-c_2), c_2),$$

wobei $M = \begin{pmatrix} m & r_1 \\ r_2 & r_3 \end{pmatrix} \in GL(2,\mathbb{Z}_p)$ ist, und Kenntnis der darin verschlüsselten Nachricht $m \in \mathbb{Z}_p$ das Produkt $M \cdot \theta_1(c_2)$ berechnet werden kann. Damit kann der Angreifer genau wie im Fall ohne Padding Instanzen des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$ erzeugen, die ausreichen, um ein $\hat{g} \in GL(2,\mathbb{Z}_p)$ mit $Inn(\hat{g}) = Inn((x\theta_1(y))^{ab})$ zu berechnen und alle folgenden Geheimtexte zu entschlüsseln.

Es bleibt zu zeigen, dass der Angreifer aus m und $C(m)$ bereits $M \cdot \theta_1(c_2)$ berechnen kann. Zunächst setzen wir $\theta_1(c_2) = \begin{pmatrix} t_1 & t_2 \\ t_3 & t_4 \end{pmatrix}$ und $Z = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix} = M \cdot \theta_1(c_2)$.

Die Funktion $Inn(g)$, wobei $g = (x,y) \in GL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$, ist Teil des öffentlichen Schlüssels des Empfängers. Wähle $X_1, X_2 \in GL(2,\mathbb{Z}_p)$ mit $X_1 \notin Z(X_2)$ und berechne

$$Inn(g)((X_1,0)) = ((x\theta_1(y)) \cdot X_1 \cdot (x\theta_1(y))^{-1}, 0)$$

und

$$Inn(g)((X_2,0)) = ((x\theta_1(y)) \cdot X_2 \cdot (x\theta_1(y))^{-1}, 0).$$

Mit $X_1, (x\theta_1(y)) \cdot X_1 \cdot (x\theta_1(y))^{-1}$ und $X_2, (x\theta_1(y)) \cdot X_2 \cdot (x\theta_1(y))^{-1}$ erhält man 2 Instanzen des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$, die ausreichen, um das spezielle Konjugationsproblem für $Inn(x\theta_1(y))$ zu lösen (siehe Abschnitt 2.7.4). Sei $H = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix}$ die Lösung dieser Instanz des SCP, d.h. $Inn(H) = Inn(x\theta_1(y))$. Dann gilt $H \in Z(x\theta_1(y))$ (siehe Abschnitt 4.3.1).

Zunächst extrahiert man $C := (x\theta_1(y))^{ab} \cdot (M \cdot \theta_1(c_2)) \cdot (x\theta_1(y))^{-ab}$ aus dem Geheimtext: Für $C(m) = (c_1, c_2)$ gilt $C = c_1 \cdot \theta_1(c_2) = (x\theta_1(y))^{ab} \cdot M\theta_1(c_2) \cdot (x\theta_1(y))^{-ab}$. Durch die Invarianz der Spur unter Konjugation erhält man $\mathrm{Spur}(C) = \mathrm{Spur}(M \cdot \theta_1(c_2)) = z_1 + z_4$. Da $Z = M \cdot \theta_1(c_2)$ gilt, erhält man die beiden Gleichungen $z_1 = mt_1 + r_1t_3$ und

$z_2 = mt_2 + r_1 t_4$. Löst man die erste Gleichung nach r_1 auf und setzt sie in die zweite Gleichung ein, so ergibt sich $z_2 = (z_1 - mt_1) \cdot \frac{t_4}{t_3} + mt_2$.
Weil H im Zentralisator von $x\theta_1(y)$ liegt, gilt $H \cdot C = (x\theta_1(y))^{ab} \cdot (H \cdot M \cdot \theta_1(c_2)) \cdot (x\theta_1(y))^{-ab}$. Die Gleichung $\text{Spur}(H \cdot C) = \text{Spur}(H \cdot M \cdot \theta_1(c_2)) = h_1 z_1 + h_2 z_3 + h_3 z_2 + h_4 z_4$ folgt wiederum aus der Invarianz der Spur unter Konjugation und wir können auch die Komponente z_3 in Abhängigkeit von z_1 darstellen:

$$\begin{aligned} z_3 &= \frac{\text{Spur}(H \cdot C) - h_1 z_1 - h_3 z_2 - h_4 z_4}{h_2} \\ &= \frac{\text{Spur}(H \cdot C) - h_1 z_1 - h_3((z_1 - mt_1) \cdot \frac{t_4}{t_3} + mt_2) - h_4(\text{Spur}(C) - z_1)}{h_2} \end{aligned}$$

Nutzt man nun die Invarianz der Determinante unter Konjugation aus, so erhält man mit

$$\begin{aligned} \det(C) &= \det(M \cdot \theta_1(c_2)) = z_1 z_4 - z_2 z_3 \\ &= z_1(\text{Spur}(C) - z_1) - ((z_1 - mt_1) \cdot \frac{t_4}{t_3} + mt_2) \cdot \\ &\quad \frac{\text{Spur}(H \cdot C) - h_1 z_1 - h_3((z_1 - mt_1) \cdot \frac{t_4}{t_3} + mt_2) - h_4(\text{Spur}(C) - z_1)}{h_2} \end{aligned}$$

eine quadratische Gleichung in z_1. Durch z_1 ist die Matrix Z bereits vollständig bestimmt. Für beide Lösungen der quadratischen Gleichung berechnet man anschließend $Z \cdot \theta_1(c_2)$. Für das richtige z_1 ergibt sich als $(1,1)$-Komponente von $Z \cdot \theta_1(c_2)$ die verschlüsselte Nachricht m. Für das falsche z_1 trifft das nur mit vernachlässigbar kleiner Wahrscheinlichkeit zu. Der Angreifer kann also mit hoher Wahrscheinlichkeit die richtige Lösung herausfinden und die Matrix $Z = M \cdot \theta_1(c_2)$ eindeutig bestimmen.

Grundversion von MOR auf $\mathbf{SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p}$: Genau wie bei Benutzung von MOR mit festem Exponenten kann ein Angreifer aus $Inn(g)$ durch Lösen des SCP in $GL(2,\mathbb{Z}_p)$ wieder ein Element $H \in GL(2,\mathbb{Z}_p)$ mit $H \in Z(x\theta_1(y))$ berechnen. Aus dem vorliegenden Geheimtext, kann ein Angreifer wieder den Wert $(x\theta_1(y))^{ab} M\theta_1(m)(x\theta_1(y))^{-ab}$ extrahieren, der einen Geheimtext der Nachricht $M\theta_1(m)$ in MOR auf $GL(2,\mathbb{Z}_p)$ darstellt. Durch den in Abschnitt 4.3.1 beschriebenen Ciphertext-Only Angriff kann die Matrix $M\theta_1(m)$ in Abhängigkeit einer Komponenten angegeben werden. Da m und damit $\theta_1(m)$ bekannt sind, kann auch die Matrix M in Abhängigkeit einer Komponenten dargestellt und die Menge der möglichen Klartexte auf eine exponentiell kleine Teilmenge des Klartextraumes eingeschränkt werden. Damit ist MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ in der Grundversion ebenfalls anfällig gegen Ciphertext-Only Angriffe. Wird das vorgeschlagene Padding-Verfahren verwendet, so liefert dieser Angriff keine Informationen über die verschlüsselte Nachricht.
Die vorgestellten Known-Plaintext Angriffe zielen darauf ab, bekannte Paare bestehend aus Klar- und zugehörigem Geheimtext zu nutzen, um den Wert $Inn((x\theta_1(y))^{ab})$ zu berechnen. Verwendet man für jede Verschlüsselung einen zufälligen Exponenten b, so ist dieser Angriff nutzlos.

4.5 Fazit

Durch die im letzten Kapitel beschriebenen Angriffe hat sich MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ für die meisten Betriebsmodi als unsicher herausgestellt. Die vorgestellten Angriffe nutzen dabei vor allem die Tatsache aus, dass sich zahlreiche Fragestellungen von MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ auf entsprechende Fragestellungen von MOR auf $SL(2, \mathbb{Z}_p)$ reduzieren lassen.
In der Grundversion von MOR konnte für einen gegebenen Geheimtext die Menge der möglichen Klartexte auf einen exponentiell kleinen Teil des Klarextraumes eingeschränkt werden. Bei Verwednung von MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ mit festem Verschlüsselungsexponenten kann aus Kenntnis von 2 Paaren bestehend aus Klar- und zugehörigem Geheimtext sogar ein zum geheimen Schlüssel äquivalenter Wert berechnet werden, der das effiziente Entschlüsseln aller folgenden Geheimtexte ermöglicht.
Einzig die Grundversion von MOR bei zusätzlichem Einsatz eines randomisierten Padding-Verfahrens hat sich als resistent gegen die vorgestellten Angriffe herausgestellt. Allerdings existiert auch für diese Variante kein Sicherheitsbeweis. Hinzu kommt, dass 2 Exponentiationen in der Gruppe der inneren Automorphismen während der Verschlüsselung und eine weitere während der Entschlüsselung berechnet werden müssen. Die Berechnung dieser Exponentiationen ist so zeitintensiv, dass MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ bei der Entschlüsselung ca. 32-mal langsamer und bei der Verschlüsselung ca. 2500-mal langsamer als RSA ist. Aus Effizienzgründen scheidet MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ damit für den Einsatz in der Praxis aus.
Damit haben sich die bekannten Betriebsmodi von MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ alle als unsicher oder ineffizient herausgestellt.

5 MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$

In [43] stellen Paeng, Kwon, Ha und Kim ein Konstruktionsprinzip für weitere Gruppen vor, auf denen das MOR System ausgeführt werden kann. Für einen kommutativen Ring mit Eins R und einen Ringautomorphismus $\Phi : R \to R$ definieren sie das semi-direkte Produkt $GL(2,R) \times_\theta \mathbb{Z}_n$ und schlagen diese zur Benutzung in MOR System vor.
Ein Schwachpunkt von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ war, dass die Abbildung θ im wesentlichen aus einem inneren Automorphismus bestand. Dadurch kann das Problem der Berechnung diskreter Logarithmen in $Inn(SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p)$ auf das Problem der Berechnung diskreter Logarithmen in $SL(2,\mathbb{Z}_p)$ und in $\mathbb{Z}_p$ reduziert werden (siehe [43]). Die in Kapitel 3 beschriebenen Angriffe auf MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ nutzen diese spezielle Wahl von θ ebenfalls aus. Bei der vorgeschlagenen Gruppe $GL(2,R) \times_\theta \mathbb{Z}_n$ wird die Abbildung θ so gewählt, dass sowohl die angesprochene Reduktion als auch die Angriffe aus Kapitel 3 nicht mehr funktionieren. Die Autoren geben jedoch keinen Beweis für die Sicherheit für MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ an.

Im folgenden Kapitel stellen wir zunächst den Vorschlag aus [43] zur Konstruktion der Gruppe $GL(2,R) \times_\theta \mathbb{Z}_n$ vor. Besonderes Augenmerk richten wir dabei auf die Anforderungen an den Ringautomorphismus Φ, die für eine Nutzbarkeit von $GL(2,R) \times_\theta \mathbb{Z}_n$ mit dem MOR System erfüllt sein müssen. Anschließend untersuchen wir die verschiedenen Betriebsmodi auf ihre Sicherheit hin. Dabei wird sich die Abbildung Φ als zentral für die Sicherheit von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ herausstellen. Abschnitt 5.5 stellt so etwas wie einen technischen Anhang für dieses Kapitel dar. In Abschnitt 2.7.4 wurde gezeigt, wie durch simultane Lösung mehrerer Instanzen des Konjugationsproblems in $GL(2,\mathbb{Z}_p)$ das spezielle Konjugationsproblem in $GL(2,\mathbb{Z}_p)$ gelöst werden kann. Wir übertragen diese Überlegungen in Abschnitt 5.5 auf den Fall, dass es sich bei der zugrundeliegeneden Struktur nicht um den Körper $\mathbb{Z}_p$ sondern um einen beliebigen kommutativen Ring mit Einselement handelt.

Sei R ein kommutativer Ring mit Eins und $\Phi : R \to R$ ein Ringautomorphismus, d.h. für alle $a, b \in R$ gilt

$$\begin{aligned} \Phi(a \cdot b) &= \Phi(a) \cdot \Phi(b) \quad \text{und} \\ \Phi(a + b) &= \Phi(a) + \Phi(b). \end{aligned}$$

Wir nehmen an, dass Φ nicht trivial ist, d.h. nicht alle Elemente auf die Null abbildet. Für Φ gilt dann ferner, dass $\Phi(0) = 0$ und $\Phi(1) = 1$.

Wir betrachten nun die (multiplikative) Gruppe $GL(2,R)$ der invertierbaren Matrizen über dem Ring R, d.h. $GL(2,R) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \,\middle|\, ad - bc \text{ ist invertierbar in } R \right\}$.

Der Ringautomorphismus Φ induziert einen Gruppenautomorphismus

$$\phi : GL(2,R) \to GL(2,R) \text{ durch}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \begin{pmatrix} \Phi(a) & \Phi(b) \\ \Phi(c) & \Phi(d) \end{pmatrix}.$$

Durch Setzen von $\theta(k) = \phi^k$ erhält man einen Homomorphismus $\theta : \mathbb{Z}_n \to Aut(GL(2,R))$, d.h.

$$\theta(k) = \phi^k : \begin{pmatrix} a & b \\ c & d \end{pmatrix} \to \begin{pmatrix} \Phi^k(a) & \Phi^k(b) \\ \Phi^k(c) & \Phi^k(d) \end{pmatrix}.$$

Wir betrachten nun das MOR System auf dem semi-direkten Produkt $GL(2,R) \times_\theta \mathbb{Z}_n$.

Konjugationen in $\mathbf{GL(2,R)} \times_\theta \mathbb{Z}_\mathbf{n}$: Seien $(x,y),(a,b) \in G \times_\theta H$. Dann gilt (siehe Abschnitt 2.9):

$$(x,y)(a,b)(x,y)^{-1} = (x\theta(y)(a)\theta(b)(x^{-1}), b).$$

Auf unsere spezielle Gruppe $G = GL(2,R) \times_\theta \mathbb{Z}_n$ und Abbildung θ angewandt bedeutet dies für $(x,y),(a,b) \in GL(2,R) \times_\theta \mathbb{Z}_n$ mit $x = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}, a = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$:

$$\begin{aligned} (x,y)(a,b)(x,y)^{-1} &= (x \cdot \phi^y(a) \cdot \phi^b(x^{-1}), b) \\ &= (x \cdot \begin{pmatrix} \Phi^y(a_1) & \Phi^y(a_2) \\ \Phi^y(a_3) & \Phi^y(a_4) \end{pmatrix} \cdot \begin{pmatrix} \Phi^b(\frac{x_4}{k}) & \Phi^b(\frac{-x_2}{k}) \\ \Phi^b(\frac{-x_3}{k}) & \Phi^b(\frac{x_1}{k}) \end{pmatrix}, b), \end{aligned}$$

wobei $k = \det x = x_1x_4 - x_2x_3$.

Wir beschäftigen uns nun mit der Frage, wie die Parameter Φ und n gewählt sein müssen, damit die Abbildung θ wohldefiniert ist.

Satz 21 *Sei $G = GL(2,R) \times_\theta \mathbb{Z}_n$ wie oben definiert. Dann gilt:*

1. $\text{ord}(\Phi) = \text{ord}(\phi)$
2. $\theta(n) = Id_{GL(2,R)} \Leftrightarrow n \equiv 0 \pmod{\text{ord}(\Phi)}$
3. *Für $(x,y),(x,\hat{y}) \in G$ gilt*

$$Inn((x,y)) = Inn((x,\hat{y})) \Leftrightarrow y \equiv \hat{y} \pmod{\text{ord}(\Phi)}.$$

4. *Die Abbildung θ ist genau dann wohldefiniert, wenn* $\text{ord}(\Phi) \mid n$.

Beweis: Aussage 1 folgt direkt aus der Definition der Abbildung ϕ und Aussage 2 direkt aus der Definition der Abbildung θ.
Seien nun $(a,b),(x,y),(x,\hat{y}) \in G$. Es gilt

$$\begin{aligned} Inn((x,y))((a,b)) &= (x\theta(y)(a)\theta(b)(x^{-1}),b) \quad \text{und} \\ Inn((x,\hat{y}))((a,b)) &= (x\theta(\hat{y})(a)\theta(b)(x^{-1}),b). \end{aligned}$$

Damit gilt $Inn((x,y)) = Inn((x,\hat{y})) \Leftrightarrow \theta(y) = \theta(\hat{y})$. Da θ ein Homomorphismus ist, folgt mit Aussage 2 die Behauptung von Aussage 3.
Zu Aussage 4. Wegen der Homomorphie von θ gilt

$$\begin{aligned} \theta(a) = \theta(a + k \cdot n) = \theta(a) \circ \theta(k \cdot n) &\Leftrightarrow \theta(k \cdot n) = Id_{GL(2,R)} \\ &\Leftrightarrow k \cdot n \equiv 0 \pmod{\text{ord}(\Phi)}. \end{aligned}$$

Die Abbildung θ ist also genau dann wohldefiniert, wenn $k \cdot n \equiv 0 \pmod{\text{ord}(\Phi)}$ für alle $k \in \mathbb{Z}$. Daraus folgt Aussage 4. □

Bemerkung 5 (Wahl von Φ) *Seien $(x,y) \in GL(2,R) \times_\theta \mathbb{Z}_n$ und $a,b \in \mathbb{Z}$. Dann ist $(x,y)^{ab} = (\hat{x}, aby \pmod n)$ für ein $\hat{x} \in GL(2,R)$. Mit MOR verschlüsselte Geheimtexte der Nachricht $(m_1,m_2) \in GL(2,R) \times_\theta \mathbb{Z}_n$ sind von der Form*

$$Inn((x,y)^{ab})((m_1,m_2)) = (\hat{x}\phi^{aby}(m_1)\phi^b(\hat{x}^{-1}),m_2).$$

Die Werte $a,b,y \in \mathbb{Z}_n$ sollten so gewählt werden, dass sie teilerfremd zur Ordnung der Abbildung ϕ sind. Ist nämlich $\text{ggt}(x,\text{ord}(\phi)) > 1$ für ein $x \in \mathbb{Z}_n$, so ist die Abbildung ϕ^x kein Generator der von ϕ erzeugten zyklischen Gruppe $<\phi>$. Hat einer der Werte $a,b,y \in \mathbb{Z}_n$ einen gemeinsamen Teiler mit $\text{ord}(\phi)$, so ist $<\phi^{aby}>$ eine (echte) Untergruppe von $<\phi>$. Dadurch wird die Menge der möglichen Geheimtexte für die Nachricht (m_1,m_2) unnötig eingeschränkt.

Um zu verhindern, dass bei der Wahl des Exponenten b bei jeder Verschlüsselung auf Teilerfremdheit zu $\text{ord}(\phi)$ getestet werden muss, sollte die Abbildung ϕ mit primer Ordnung gewählt werden.

5.1 Das DLP in $Inn(GL(2,R) \times_\theta \mathbb{Z}_n)$ und das DLP in $<\Phi>$

Wir zeigen zunächst, dass für $g = (x,y) \in GL(2,R) \times_\theta \mathbb{Z}_n$ aus einem gegebenen Automorphismus $Inn(g)$ die Automorphismen ϕ^y bzw. Φ^y effizient berechnet werden können.

Zunächst nutzen wir aus, dass die Abbildung Φ als (nicht-trivialer) Ringautomorphismus die Null wieder auf die Null und die Eins wieder auf die Eins abbildet und dass $\phi^0 = Id$ gilt. Für eine unimodulare Matrix m (d.h. eine Matrix, die nur Einträge aus Null und Eins hat) folgt, dass $\phi^y(m) = m$ gilt, und es ergibt sich

$$\begin{aligned} Inn(g)((m,0)) = (x,y)(m,0)(x,y)^{-1} &= (x \cdot \phi^y(m) \cdot \phi^0(x^{-1}), 0) \\ &= (x \cdot m \cdot x^{-1}, 0). \end{aligned}$$

Durch Berechnung des Funktionswerts von $(m,0)$ unter $Inn(g)$ erhält man mit $m, x \cdot m \cdot x^{-1}$ also eine Instanz des Konjugationsproblems in $GL(2,R)$. Durch simultanes Lösen der beiden Instanzen $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $x \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot x^{-1}$ und $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, $x \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot x^{-1}$ kann ein $\hat{x} \in GL(2,R)$ berechnet werden mit $Inn(\hat{x}) = Inn(x)$ (siehe Abschnitt5.5).

Wählt man nun $m \in GL(2,R)$ beliebig, so erhält man

$$Inn(g)((m,0)) = (x,y)(m,0)(x,y)^{-1} = (x \cdot \phi^y(m) \cdot x^{-1}, 0).$$

Nutzt man nun die Kenntnis von $\hat{x}$ aus, so ergibt sich

$$Inn(\hat{x}^{-1})(x \cdot \phi^y(m) \cdot x^{-1}) = (\hat{x}^{-1}x) \cdot \phi^y(m) \cdot (\hat{x}^{-1}x)^{-1} = \phi^y(m).$$

Aus Kenntnis von $Inn(g)$ kann also insbesondere die Funktionen ϕ^y und damit auch Φ^y berechnet werden. Wendet man das selbe Verfahren auf die Funktion $Inn(g^a)$ an, so können ϕ^{ay} bzw. Φ^{ay} ebenfalls berechnet werden. Da $Inn(g)$ und $Inn(g^a)$ beide Teil des öffentlichen Schlüssels sind, ist die Schwierigkeit des Diskreter Logarithmus Problems in $<\Phi>$ eine notwendige Bedingung für die Sicherheit von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$. Ist die Berechnung diskreter Logarithmen effizient möglich in $<\Phi>$, so kann mit $a \pmod{\mathrm{ord}(\Phi)}$ ein Teil des geheimen Schlüssels a berechnet werden.

5.2 Das DLP in $Inn(GL(2,R) \times_\theta \mathbb{Z}_n)$ und das SCP in $GL(2,R) \times_\theta \mathbb{Z}_n$

Ziel des folgenden Abschnitts ist es zu zeigen, dass mit $a \pmod{\text{ord}(\Phi)}$ Teile des geheimen Schlüssels berechnet werden können, sofern das spezielle Konjugationsproblem effizient lösbar ist in der Gruppe $GL(2,R) \times_\theta \mathbb{Z}_n$.

Dazu schauen wir uns zunächst an, welche Struktur das Zentrum im Ring $M(2,R)$ und der Gruppe $GL(2,R)$ hat.

Lemma 4 *Sei R ein kommutativer Ring mit Eins. Dann ist $Z(M(2,R)) = \{c \cdot Id \mid c \in R\}$ und $Z(GL(2,R)) = \{c \cdot Id \mid c \in R, c \text{ invertierbar}\}$.*

Beweis: Sei $Z = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix} \in Z(GL(2,R))$. Mit $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \in GL(2,R)$ erhält man die Gleichung $Z \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot Z$. Durch Koeffizientenvergleich folgt, dass $z_2 = z_3$ und $z_1 = z_4$ gelten. Analog folgt mit $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \in GL(2,R)$, dass $Z \cdot \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \cdot Z$ und damit $z_2 = z_3 = 0$ gelten. Die Matrix Z ist also von der Form $c \cdot Id$ mit $c \in R$. Da der Ring R als kommutativ vorausgesetzt wurde, gilt $Z(M(2,R)) = \{c \cdot Id \mid c \in R\}$ und $Z(GL(2,R)) = \{c \cdot Id \mid c \in R\} \cap GL(2,R) = \{c \cdot Id \mid c \in R, c \text{ invertierbar}\}$. □

Als nächstes beschäftigen wir uns mit der Frage, in welcher Relation zwei Elemente $g = (x,y), \hat{g} = (\hat{x},\hat{y}) \in GL(2,R) \times_\theta \mathbb{Z}_n$ stehen müssen, damit sie den selben inneren Automorphismus definieren.

Satz 22 *Sei $G = GL(2,R) \times_\theta \mathbb{Z}_n$ mit θ wie oben definiert. Seien ferner $(x,y), (\hat{x},\hat{y}) \in G$. Dann gilt $Inn((x,y)) = Inn((\hat{x},\hat{y}))$ genau dann, wenn $\hat{x} = x \cdot \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix}$ für einen Fixpunkt c von Φ und $y \equiv \hat{y} \pmod{\text{ord}(\Phi)}$ erfüllt sind.*

Beweis: Seien $g = (x,y), \hat{g} = (\hat{x},\hat{y}) \in G$ mit $Inn(g) = Inn(\hat{g})$. Betrachte $Inn(g)((a,0))$ und $Inn(\hat{g})((a,0))$ für eine unimodulare Matrix $a \in GL(2,R)$. Dann gilt

$$(x \cdot a \cdot x^{-1}, 0) = Inn(g)((a,0)) = Inn(\hat{g})((a,0)) = (\hat{x} \cdot a \cdot \hat{x}^{-1}, 0).$$

Da diese Bedingung insbesondere für die beiden Matrizen $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \in GL(2,R)$ erfüllt ist, muss $Inn(x) = Inn(\hat{x})$ gelten (siehe Abschnitt 5.5). Es gibt also ein $z_1 \in Z(GL(2,R))$ mit $\hat{x} = x \cdot z_1$, d.h. $\hat{x} = x \cdot \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix}$ für ein $c \in R$.

Da $Inn(g) = Inn(\hat{g})$, gilt für alle $(a,b) \in GL(2,R) \times_\theta \mathbb{Z}_n$

$$\begin{aligned} Inn(g)((a,b)) &= Inn(\hat{g})((a,b)) \\ \Leftrightarrow x \cdot \phi^y(a) \cdot \phi^b(x^{-1}) &= \hat{x} \cdot \phi^{\hat{y}}(a) \cdot \phi^b(\hat{x}^{-1}) \\ \Leftrightarrow x \cdot \phi^y(a) \cdot \phi^b(x^{-1}) &= \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix} \cdot x \cdot \phi^{\hat{y}}(a) \cdot \phi^b(x^{-1}) \cdot \phi^b \begin{pmatrix} c^{-1} & 0 \\ 0 & c^{-1} \end{pmatrix} \\ \Leftrightarrow \phi^y(a) &= \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix} \cdot \phi^{\hat{y}}(a) \cdot \phi^b \begin{pmatrix} c^{-1} & 0 \\ 0 & c^{-1} \end{pmatrix}. \end{aligned} \tag{2}$$

Für beliebige $b_1, b_2 \in \mathbb{Z}_n$ folgt aus $Inn(g)((a,b_i)) = Inn(\hat{g})((a,b_i))$ für $i = 1,2$

$$\begin{aligned} \phi^y(a) &= \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix} \cdot \phi^{\hat{y}}(a) \cdot \phi^{b_i} \begin{pmatrix} c^{-1} & 0 \\ 0 & c^{-1} \end{pmatrix} \\ &= \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix} \cdot \phi^{\hat{y}}(a) \cdot \begin{pmatrix} \Phi^{b_i}(c^{-1}) & 0 \\ 0 & \Phi^{b_i}(c^{-1}) \end{pmatrix} \end{aligned} \tag{3}$$

und damit $\Phi^{b_1}(c^{-1}) = \Phi^{b_2}(c^{-1})$, d.h. $c = \Phi^{b_1 - b_2}(c)$. Für $b_1 - b_2 = 1$ folgt, dass c ein Fixpunkt von Φ sein muss. Gleichung 3 vereinfacht sich damit zu $\phi^y(a) = \phi^{\hat{y}}(a)$ für alle $a \in GL(2,R)$, d.h. es gilt $\phi^y = \phi^{\hat{y}}$ und damit $y \equiv \hat{y} \pmod{\text{ord}(\phi)}$.
Die Rückrichtung folgt, wenn man bei den Umformungen von Gleichung 2 ausnutzt, dass c ein Fixpunkt von Φ und $y \equiv \hat{y} \pmod{\text{ord}(\phi)}$ ist. □

Die für unsere Zwecke wesentliche Aussage des letzten Satzes ist die Tatsache, dass für zwei Elemente $g = (x,y), \hat{g} = (\hat{x},\hat{y}) \in GL(2,R) \times_\theta \mathbb{Z}_n$, die den selben inneren Automorphismus definieren, die Bedingung $y \equiv \hat{y} \pmod{\text{ord}(\Phi)}$ erfüllt sein muss. Da in der (additiven) Gruppe $\mathbb{Z}_n$ die Berechnung diskreter Logarithmen effizient möglich ist, kann der Wert $a \bmod \text{ord}(\Phi)$ berechnet werden, sofern das spezielle Konjugationsproblem in $GL(2,R) \times_\theta \mathbb{Z}_n$ effizient lösbar ist.

Satz 23 *Sei* $\text{ord}(\Phi)$ *prim. Ist das SCP leicht in* $GL(2,R) \times_\theta \mathbb{Z}_n$*, so kann* $a \bmod \text{ord}(\Phi)$ *aus den bekannten Abbildungen* $Inn(g)$ *und* $Inn(g^a)$ *effizient berechnet werden.*

Beweis: Für $g = (x,y)$ ist $g^a = (\hat{x}, ay \bmod n)$ für ein $\hat{x} \in GL(2,R)$. Durch Lösen der Instanzen $Inn(g)$ und $Inn(g^a)$ des SCP erhält man $h = (h_1,h_2), \hat{h} = (\hat{h}_1,\hat{h}_2) \in G$ mit $Inn(g) = Inn(h)$ und $Inn(g^a) = Inn(\hat{h})$.
Mit Satz 22 folgt, dass $y \equiv h_2 \pmod{\text{ord}(\Phi)}$ und $a \cdot y \equiv \hat{h}_2 \pmod{\text{ord}(\Phi)}$. Der gesuchte Wert ergibt sich als $a \equiv \hat{h}_2 \cdot h_2^{-1} \pmod{\text{ord}(\Phi)}$. □

Der Satz gilt auch, wenn die Abbildung Φ keine prime Ordnung hat, der Wert $y \in \mathbb{Z}_n$ aber teilerfremd zu $\text{ord}(\Phi)$ gewählt wurde (wie in Bemerkung 5 empfohlen). Im Fall $d = \text{ggt}(y, \text{ord}(\Phi)) > 1$ hat die Gleichung $\hat{h_2} \equiv a \cdot y \pmod{\text{ord}(\Phi)}$ genau d Lösungen.

5.3 Sicherheitsanalyse der verschiedenen Betriebsmodi

In diesem Abschnitt werden wir die Sicherheit der verschiedenen Betriebsmodi von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ untersuchen. Dabei werden folgende Varianten der Grundversion betrachtet:

- Einschränkung der Abbildung $Inn(g)$ auf $GL(2,R) \times_\theta \{0\}$
- Benutzung des (während der Verschlüsselung gewählten) Exponenten b für die Verschlüsselung mehrerer Klartexte
- Probabilistische Einbettung der zu verschlüsselnden Nachricht $m \in R$ in $GL(2,R)$ durch das in [42] vorgeschlagene Padding-Verfahren (siehe Abschnitt 4.2)

Wie in Abschnitt 5.1 gezeigt wurde, kann ein Angreifer bei MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ aus dem öffentlichen Schlüssel $Inn(g), Inn(g^a)$ für $g = (x,y) \in GL(2,R) \times_\theta \mathbb{Z}_n$ die Abbildungen ϕ^y und ϕ^{ay} berechnen. Für die Sicherheit von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ ist es also notwendig, dass die Berechnung diskreter Logarithmen in der Gruppe $<\phi>$ nicht effizient möglich ist. Ansonsten könnte mit $a \bmod \mathrm{ord}(\phi)$ ein Teil des geheimen Schlüssels a berechnet werden.

Aus der Abbildung $Inn(g^b)$, die Teil des Geheimtextes ist, kann analog ϕ^{by} berechnet werden. Im folgenden untersuchen wir die Frage, welche Bedeutung die Schwierigkeit des Computational Diffie-Hellman Problems in $<\phi>$ für die Sicherheit von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ hat.

MOR auf $\mathbf{GL(2,R) \times_\theta \mathbb{Z}_n}$ mit eingeschränktem Inn(g): Die Abbildung $Inn(g)$ sei auf $GL(2,R) \times_\theta \{0\}$ eingeschränkt. Für $g = (x,y) \in GL(2,R) \times_\theta \mathbb{Z}_n$ und $a, b \in \mathbb{Z}$ ist $g^{ab} = (x,y)^{ab} = (\hat{x}, aby \bmod n)$ für ein $\hat{x} \in GL(2,R)$. Geheimtexte der Nachricht $(m,0) \in GL(2,R) \times_\theta \{0\}$ sind von der Form

$$Inn(g^{ab})((m,0)) = (\hat{x} \cdot \phi^{aby}(m) \cdot \hat{x}^{-1}, 0).$$

Die Verschlüsselungsfunktion entspricht also der Funktion

$$Inn(\hat{x}) \circ \phi^{aby} : GL(2,R) \to GL(2,R).$$

Da Spur und Determinante invariant unter Konjugation sind, können aus dem Geheimtext $\mathrm{Spur}(\phi^{aby}(m))$ und $\det(\phi^{aby}(m))$ berechnet werden. Hier zeigt sich die Bedeutung, die der Abbildung ϕ für die Sicherheit des Systems zukommt. Gelingt es einem Angreifer, die Abbildung ϕ^{aby} zu berechnen, so sind bei MOR auf $GL(2,R) \times_\theta \{0\}$ in allen Modi, d.h. mit und ohne festen Exponenten b und mit und ohne Padding, die selben Angriffe möglich, die in Kapitel 4.3 für MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ vorgestellt wurden.

Schränkt man bei MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ die Funktion $Inn(g)$ auf $SL(2,\mathbb{Z}_p) \times_\theta \{0\}$ ein, so gilt für $g = (x,y)$, dass $Inn(g) = Inn(x\theta_1(y))$ und $Inn(g^a) = Inn((x\theta_1(y))^a)$. Der

Wert $y \in \mathbb{Z}_p$ geht lediglich in das konjugierende Element $x\theta_1(y)$ ein. Ist $x\theta_1(y)$ durch die Wahl von $g = (x, y)$ fixiert, wird y für die Berechnung von $Inn(g^a)$ nicht mehr benötigt. Das hat zur Folge, dass zur Berechnung von $Inn(g^a)$ in $SL(2, \mathbb{Z}_p)$ anstatt in $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ gerechnet werden kann, was die Effizienz merklich steigert. Schränkt man hingegen bei MOR auf $GL(2, R) \times_\theta \mathbb{Z}_n$ die Funktion $Inn(g)$ auf $GL(2, R) \times_\theta \{0\}$ ein, so gilt für $g = (x, y) \in GL(2, R) \times_\theta \mathbb{Z}_n$, dass $Inn(g) = Inn(x) \circ \phi^y$ und $Inn(g^a) = Inn(\hat{x}) \circ \phi^{ay}$, wobei $\hat{x} = \prod_{i=i}^{a-1} \phi^{iy}(x)$. Die Aufwändigkeit der Berechnung von $Inn(g^a)$ ist damit unabhängig davon, ob die Abbildung $Inn(g)$ auf $GL(2, R) \times_\theta \{0\}$ eingeschränkt ist oder nicht.
Die Einschränkung der Funktion $Inn(g)$ auf $GL(2, R) \times_\theta \{0\}$ eröffnet also einerseits weitere Angriffsmöglichkeiten. Dies ist insbesondere der Fall, wenn Informationen über ϕ^{aby} berechnet werden können. Andererseits hat die Einschränkung jedoch keinen positiven Einfluss auf die Effizienz des Verfahrens. Aus kryptologischer Sicht ist eine Eischränkung von $inn(g)$ also nicht empfehlenswert.

MOR auf $\mathbf{GL(2, R) \times_\theta \mathbb{Z}_n}$ mit festem Exponenten: Wir betrachten nun MOR auf $GL(2, R) \times_\theta \mathbb{Z}_n$, wenn der Exponent b einmal gewählt und für mehrere Verschlüsselungen verwendet wird. Wir nehmen im Folgenden an, dass der Angreifer die Abbildung ϕ^{aby} kennt, und stellen einen Chosen-Ciphertext Angriff vor, der es einem solchen Angreifer erlaubt, beliebige Geheimtexte zu entschlüsseln. Wie üblich bekommt der Angreifer zunächst den zu entschlüsselnden Geheimtext $C \in GL(2, R) \times_\theta \mathbb{Z}_n$. Anschließend darf er einem Entschlüsselungsorakel beliebige Geheimtexte $C_i \neq C$ vorlegen und erhält die dazugehörigen Klartexte als Antwort.

Da die Abbildungen $Inn(g)$ für alle $g \in G$ Automorphismen sind, ist die Verschlüsselung von MOR auf $GL(2, R) \times_\theta \mathbb{Z}_n$ eine bijektive Abbildung. Jedes $D \in GL(2, R) \times_\theta \mathbb{Z}_n$ ist also ein gültiger Geheimtext einer (eventuell unbekannten) Nachricht $M \in GL(2, R) \times_\theta \mathbb{Z}_n$.
Sei $g = (x, y) \in GL(2, R) \times_\theta \mathbb{Z}_n$ und $(x, y)^{ab} = (\hat{x}, aby \bmod n)$ für ein $\hat{x} \in GL(2, R)$. Der vorgestellte Chosen-Ciphertext Angriff ist zweistufig. In der ersten Stufe stellt der Angreifer seine Orakelanfragen so, dass er ein $\bar{x} \in GL(2, R)$ berechnen kann mit $Inn(\hat{x}) = Inn(\bar{x})$. In der zweiten Stufe stellt er zwei Anfragen an das Orakel, die vom vorgegebenen Geheimtext C abhängen und es ihm erlauben, diesen zu entschlüsseln.

Stufe 1: Für jedes $D \in GL(2, R)$ ist das Element $(D, 0) \in GL(2, R) \times_\theta \mathbb{Z}_n$ ein gültiger Geheimtext der (eventuell unbekannten) Nachricht $(M, 0) \in GL(2, R) \times_\theta \mathbb{Z}_n$:

$$(D, 0) = (\hat{x} \cdot \phi^{aby}(M) \cdot \hat{x}^{-1}, 0)$$

Sendet der Angreifer $(D, 0)$ an das Orakel, so erhält er den Klartext $(M, 0)$ und kann $\phi^{aby}(M)$ berechnen. Die Werte $\phi^{aby}(M)$ und $D = \hat{x} \cdot \phi^{aby}(M) \cdot \hat{x}^{-1}$ bilden eine Instanz des Konjugationsproblems in $GL(2, R)$. Durch Wiederholung dieses Vorgehens kann ein Angreifer mehrere simultane Instanzen erzeugen. Durch Lösung dieser Instanzen kann das spezielle Konjugationsproblem in $GL(2, R)$ gelöst werden, d.h. ein Wert $\bar{x} \in GL(2, R)$ mit $Inn(\hat{x}) = Inn(\bar{x})$ berechnet werden (siehe Abschnitt 5.5).

Da $GL(2,R) \times_\theta \{0\}$ isomorph zu $GL(2,R)$ ist und das (spezielle) Konjugationsproblem effizient lösbar in $GL(2,R)$ ist, wäre es denkbar, dass das Orakel die Beantwortung von Anfragen mit Null als zweiter Komponente verweigert. In diesem Fall sendet der Angreifer zwei Anfragen (D_1, i) und (D_2, i) mit identischer zweiter Komponente:

$$(D_1, i) = (\hat{x} \cdot \phi^{aby}(M_1) \cdot \phi^i(\hat{x}^{-1}), i)$$
$$(D_2, i) = (\hat{x} \cdot \phi^{aby}(M_2) \cdot \phi^i(\hat{x}^{-1}), i)$$

Aus den Klartexten $(M_1, i), (M_2, i) \in GL(2,R) \times_\theta \mathbb{Z}_n$ und Abbildung ϕ^{aby} erhält der Angreifer nun mit $\phi^{aby}(M_1 \cdot M_2^{-1})$ und $D_1 \cdot (D_2)^{-1} = \hat{x} \cdot \phi^{aby}(M_1 \cdot M_2^{-1}) \cdot \hat{x}^{-1}$ ebenfalls eine Instanz des Konjugationsproblems in $GL(2,R)$.

Stufe 2: Sei $C = (C_1, C_2)$ der zu entschlüsselnde Geheimtext. Wir wissen, dass $\bar{x}$ von der Form $\bar{x} = \hat{x} \cdot z$ mit $z \in Z(GL(2,R))$ ist. Durch Berechnung von

$$\begin{aligned} \bar{x}^{-1} \cdot C_1 \cdot \phi^{C_2}(\bar{x}) &= \bar{x}^{-1} \cdot (\hat{x} \cdot \phi^{aby}(M) \cdot \phi^{C_2}(\hat{x})) \cdot \phi^{C_2}(\bar{x}) \\ &= (\hat{x} \cdot z)^{-1} \cdot (\hat{x} \cdot \phi^{aby}(M) \cdot \phi^{C_2}(\hat{x})) \cdot \phi^{C_2}(\hat{x} \cdot z) \\ &= z^{-1} \cdot \phi^{aby}(M) \cdot \phi^{C_2}(z) \end{aligned}$$

erhält der Angreifer den Wert $\phi^{aby}(M) \cdot z^{-1} \cdot \phi^{C_2}(z)$. Er stellt nun eine geschickte Anfrage an das Entschlüsselungsorakel, aus der er den Wert $z^{-1} \cdot \phi^{C_2}(z)$ berechnen kann. Dazu wählt er $C_3 \in GL(2,R)$ mit $C_3 \neq C_1$ und stellt (C_3, C_2) als Frage an das Orakel. Sei $\widehat{M}$ die Antwort des Orakels. Dann ergibt sich der gesuchte Wert $z^{-1} \cdot \phi^{C_2}(z)$ durch

$$\begin{aligned} C_3 \cdot (\phi^{C_2}(\bar{x}) \cdot \phi^{aby}(\widehat{M}^{-1}) \cdot \bar{x}^{-1}) &= (\hat{x} \cdot \phi^{aby}(\widehat{M}) \cdot \phi^{C_2}(\hat{x}^{-1})) \cdot (\phi^{C_2}(\bar{x}) \cdot \phi^{aby}(\widehat{M}^{-1}) \cdot \bar{x}^{-1}) \\ &= \hat{x} \cdot \phi^{aby}(\widehat{M}) \cdot \phi^{C_2}(\hat{x}^{-1}) \cdot \phi^{C_2}(\hat{x} \cdot z) \cdot \phi^{aby}(\widehat{M}^{-1}) \cdot (\hat{x} \cdot z)^{-1} \\ &= z^{-1} \cdot \phi^{C_2}(z). \end{aligned}$$

Damit kann der Wert $\phi^{aby}(M)$ berechnet werden. Reicht die Kenntnis von ϕ^{aby} nicht aus, um die verschlüsselte Nachricht M aus $\phi^{aby}(M)$ zu berechnen, so kann das Entschlüsselungsorakels benutzt werden, um Urbilder von ϕ^{aby} zu berechnen. Um das Urbild von $\phi^{aby}(M)$ berechnen zu können, sendet der Angreifer

$$(D, 0) = (\bar{x} \cdot \phi^{aby}(M) \cdot \bar{x}^{-1}, 0) = (\hat{x} \cdot \phi^{aby}(M) \cdot \hat{x}^{-1}, 0)$$

als Anfrage an das Entschlüsselungsorakel. Die Antwort des Orakels entspricht der gesuchten Nachricht M. Beantwortet das Orakel keine Anfragen mit Null als zweiter Komponente, so kann der Wert $\bar{x} \cdot \phi^{aby}(M) \cdot \bar{x}^{-1}$ dargestellt werden als $\bar{x} \cdot \phi^{aby}(M) \cdot \bar{x}^{-1} = E_1 \cdot E_2$ mit $E_1, E_2 \in GL(2,R)$. Der Angreifer sendet dann (E_1, i) und (E_2, i) als Anfragen an das Orakel. Sind F_1 und F_2 die Antworten des Orakels, so ergibt sich M als $M = F_1 \cdot F_2$ (siehe auch Stufe 1 für ein ähnliches Argument).

MOR auf $\mathbf{GL(2,R)} \times_\theta \mathbb{Z}_\mathbf{n}$ mit festem Exponenten und Padding: Benutzt man zusätzlich das Padding-Verfahren aus [42], so muss der vorgestellte Angriff angepasst werden: In Abschnitt 4.3.2 wurde gezeigt, dass für MOR auf $SL(2,\mathbb{Z}_p)$ mit festem Exponenten und Padding zwei Paare bestehend aus Klar- und zugehörigem Geheimtext ausreichen, um die Abbildung $Inn(g^{ab})$ zu berechnen. Die selbe Technik kann in Stufe 1 des Angriffes benutzt werden, um ein Element $\bar{x}$ mit $Inn(\hat{x}) = Inn(\bar{x})$ zu berechnen.

Der erste Schritt von Stufe 2 funktioniert auch dann noch, wenn das probabilistische Padding-Verfahren benutzt wird, d.h. der Angreifer kann den Wert $\phi^{aby}(M) \cdot z^{-1} \cdot \phi^{C_2}(z)$ berechnen. Der folgende Schritt muss wieder angepasst werden. Auf die Anfrage (C_3, C_2) antwortet das Entschlüsselungsorakel nur mit der $(1,1)$-Komponente der Matrix $\widehat{M}$. Die übrigen Einträge der Matrix $\widehat{M}$ sind dem Angreifer aber nicht bekannt. Da $Z(GL(2,R)) = \{c \cdot Id \mid c \in R, c \text{ invertierbar}\}$ gilt, muss der Wert $z^{-1} \cdot \phi^{C_2}(z)$ von der Form $z^{-1} \cdot \phi^{C_2}(z) = \begin{pmatrix} r & 0 \\ 0 & r \end{pmatrix}$ für ein invertierbares $r \in R$ sein. Insbesondere gilt $z^{-1} \cdot \phi^{C_2}(z) \in Z(GL(2,R))$. Für $\widehat{M} = \begin{pmatrix} \widehat{m}_1 & \widehat{m}_2 \\ \widehat{m}_3 & \widehat{m}_4 \end{pmatrix}$ erhält man

$$\begin{aligned} \bar{x}^{-1} \cdot C_3 \cdot \phi^{aby}(\bar{x}) &= (z^{-1}\hat{x}^{-1}) \cdot (\hat{x}\phi^{aby}(\widehat{M})\phi^{C_2}(\hat{x}^{-1})) \cdot (\phi^{C_2}(\hat{x}z)) \\ &= \phi^{aby}(\widehat{M}) \cdot z^{-1} \cdot \phi^{C_2}(z) \\ &= \begin{pmatrix} r \cdot \widehat{m}_1 & r \cdot \widehat{m}_2 \\ r \cdot \widehat{m}_3 & r \cdot \widehat{m}_4 \end{pmatrix}. \end{aligned}$$

Der Angreifer erhält den Wert $\widehat{m}_1$, wenn er (C_3, C_2) als Anfrage an das Orakel sendet. Kann r nicht aus $\widehat{m}_1$ und $r \cdot \widehat{m}_1$ berechnet werden, so muss dieser Schritt mit einem anderen Wert für C_3 wiederholt werden.

Die Berechnung von M aus $\phi^{aby}(M)$ funktioniert auch dann, wenn das probabilistische Padding-Verfahren benutzt wird. In diesem Fall muss das Urbild allerdings komponentenweise berechnet werden. Um das Urbild von $\phi^{aby}(M) = \begin{pmatrix} \Phi^{aby}(m_1) & \Phi^{aby}(m_2) \\ \Phi^{aby}(m_3) & \Phi^{aby}(m_4) \end{pmatrix}$ zu berechnen, wählt der Angreifer Elemente $D_i \in GL(2,R)$ $(1 \leq i \leq 4)$ so, dass die $(1,1)$-Komponente von D_i gleich $\phi^{aby}(m_i)$ ist. Er sendet dann $(\bar{x} \cdot D_i \cdot \bar{x}^{-1}, 0)$ für $1 \leq i \leq 4$ als Anfragen an das Orakel. Die $(1,1)$-Komponente der Antwort auf Anfrage $(\bar{x} \cdot D_i \cdot \bar{x}^{-1}, 0)$ ist gleich m_i.

5.4 Fazit

Die effizienten Modi von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ haben sich als höchstens so sicher wie das Computational Diffie-Hellman Problem in der vom Ringautomorphismus Φ erzeugten zyklischen Gruppe $<\Phi>$ erwiesen. Ist die Berechnung diskreter Logarithmen in $<\Phi>$ effizient möglich, so kann sogar der benutzte geheime Schlüssel teilweise berechnet werden. Das Computational Diffie-Hellman Problem und das Diskreter Logarithmus Problem in der Gruppe $<\Phi>$ sind beide leichter als das Diskreter Logarithmus Problem in der Gruppe $Inn(GL(2,R) \times_\theta \mathbb{Z}_n)$. Genau wie bei MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ hat sich bei MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ das DLP in $Inn(GL(2,R) \times_\theta \mathbb{Z}_n)$ als notwendig aber nicht als hinreichend für die Sicherheit des Verfahrens herausgestellt. Das Vertrauen in MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ wird außerdem durch die Tatsache geschwächt, dass es sich bei den vorgestellten Angriffen um generische Angriffe handelt, d.h. sie sind für jede Wahl des Rings R durchführbar. Für konkrete Wahlen von R kann es aber durchaus noch stärkere Angriffe geben, die bestimmte Eigenschaften des benutzten Rings ausnutzen.

Durch die beschriebenen Angriffe haben sich viele der betrachteten Betriebsmodi von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ als unsicher erwiesen. Bei den Modi, die den vorgestellten Angriffen standhalten, müssen bei jedem Ver- und Entschlüsselungsvorgang Potenzen in $Inn(G)$ berechnet werden. Damit scheiden diese Modi aus Effizienzgründen als Alternative zu den existierenden Kryptoverfahren aus. Wie schon bei MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ sind damit alle Betriebsmodi unsicher oder ineffizient.
Da MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ höchstens so sicher ist wie das DLP in der Gruppe $<\Phi>$, stellt sich die Frage, wie sich der zur Konstruktion und Durchführung von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ notwendige Aufwand rechtfertigen lässt. Ein ElGamal-ähnliches Verfahren auf der Gruppe $<\Phi>$ wäre wesentlich effizienter als MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ und beweisbar sicher.

5.5 Die Konjugationsprobleme in $GL(2,R)$

Der folgende Abschnitt ist eine Art technischer Anhang des aktuellen Kapitels. Er schließt inhaltlich nicht an die vorherigen Abschnitte dieses Kapitels an. Vielmehr stellt er das in den letzten Abschnitten bereits verwendete Rüstzeug für die Durchführung der beschriebenen Angriffe zur Verfügung.

Wir nehmen im Folgenden an, dass R ein kommutativer Ring mir Eins ist. Ziel dieses Abschnittes ist es, Lösungen für die Kojugationsprobleme in $GL(2,R)$ anzugeben. Wir führen das Lösen der Konjugationsprobleme in $GL(2,R)$ wieder auf das Lösen linearer homogener Gleichungssysteme in R zurück. In den folgenden Unterabschnitten werden zunächst Konstruktionen für die Berechnung von Lösungen von linearen Gleichungen und linearen Gleichungssystemen über R angegeben. Die vorgestellten Lösungsmethoden nutzen wir anschließend zur Lösung des Konjugationsproblems in $GL(2,R)$. Wir zeigen, dass das spezielle Konjugationsproblem in $GL(2,R)$ durch simultane Lösung mehrerer Instanzen des Konjugationsproblems in $GL(2,R)$ gelöst werden kann (vergleiche auch mit Abschnitt 2.7.4, in dem ein ähnliches Ergebnis für die Gruppe $GL(2,\mathbb{Z}_p)$ gezeigt wurde). Der Fokus unserer Betrachtungen liegt dabei nicht auf dem Auffinden aller Lösungen. Je nach betrachtetem Ring kann es weitere Lösungen geben.

5.5.1 Homogene lineare Gleichungen über Ringen

Wir betrachten zunächst homogene lineare Gleichungen der Form

$$a_1x_1 + \cdots + a_nx_n = 0$$

mit $a_1, \ldots a_n \in R$.

Die triviale Lösung $x_1 = \cdots = x_n = 0$ erfüllt jede homogene lineare Gleichung. Durch $x_i = \prod_{1\leq j\leq n, j\neq i} a_j$ für $1 \leq i \leq n-1$ und $x_n = -(n-1) \cdot \prod_{1\leq j\leq n-1} a_j$ lässt sich auch leicht eine nicht-triviale Lösung angeben. Der folgende Satz sagt aus, dass man aus einer gegebenen Lösung leicht weitere Lösungen erzeugen kann.

Satz 24 *Sei das Tupel $(y_1, \ldots, y_n) \in R^n$ eine Lösung von $a_1x_1 + \cdots + a_nx_n = 0$. Dann ist auch*

$$(\hat{y}_1, \ldots, \hat{y}_n) = (y_1, \ldots, y_n) + \begin{pmatrix} 0 & t_{1,2} & t_{1,3} & \cdots & t_{1,n} \\ -t_{1,2} & 0 & t_{2,3} & & t_{2,n} \\ -t_{1,3} & -t_{2,3} & 0 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & t_{n-1,n} \\ -t_{1,n} & -t_{2,n} & \cdots & -t_{n-1,n} & 0 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix}$$

für beliebige $t_{i,j} \in R$ $(1 \leq i < j \leq n)$ eine Lösung von $a_1x_1 + \cdots + a_nx_n = 0$.

Beweis: Die Aussage folgt direkt durch Einsetzen und Umsortieren der Terme:

$$\begin{aligned}\sum_{i=1}^{n} a_i \cdot \hat{y}_i &= \sum_{i=1}^{n} a_i \cdot y_i + \sum_{1 \le i < j \le n} t_{i,j} \cdot (a_i \cdot a_j - a_j \cdot a_i) \\ &= \sum_{i=1}^{n} a_i \cdot y_i = 0\end{aligned}$$

□

Um die Kojugationsprobleme über $GL(2,R)$ zu lösen sind vor allem die Fälle $n = 2, 3$ wichtig, die wir hier noch einmal hervorheben möchten. Zunächst beschäftigen wir uns mit dem Fall $n = 2$, d.h. mit Gleichungen der Form

$$a_1 x_1 + a_2 x_2 = 0.$$

Als spezielle Lösung erhält man $x_1 = a_2$ und $x_2 = -a_1$. Satz 24 liefert mit $x_1 = x_2 = 0$ als spezieller Lösung die Lösungen $\hat{x}_1 = a_2 \cdot r$ und $\hat{x}_2 = -a_1 \cdot r$ für $r \in R$.

Für den Fall $n = 3$ hat die betrachtete Gleichung die Form

$$a_1 x_1 + a_2 x_2 + a_3 x_3 = 0.$$

Diese Gleichung hat die Lösung $x_1 = a_2 a_3$, $x_2 = a_1 a_3$ und $x_3 = -2t a_1 a_2$. Satz 24 liefert mit $x_1 = x_2 = x_3 = 0$ die Lösungen $(\hat{x}_1, \hat{x}_2, \hat{x}_3) = r \cdot (a_2, -a_1, 0) + s \cdot (a_3, 0, -a_1) + t \cdot (0, a_3, -a_2)$ für $r, s, t \in R$.

Ist R ein Körper, so sind dies bereits alle Lösungen. Für $n = 2$ hat in diesem Fall der Lösungsraum die Basis $(a_2, -a_1)$ und für $n = 3$ die Basis $(a_2, -a_1, 0), (a_3, 0, -a_1)$. Enthält R Nullteiler, so kann es weitere Lösungen geben. Ist R ein unendlicher Integritätsbereich, so kann es ebenfalls weitere Lösungen geben. In einem euklidischen Ring sind etwa $x_1 = r \cdot \frac{a_2}{\mathrm{ggt}(a_1,a_2)}$ und $x_2 = r \cdot \frac{-a_1}{\mathrm{ggt}(a_1,a_2)}$ Lösungen von $a_1 x_1 + a_2 x_2 = 0$. Für $\mathrm{ggt}(a_1, a_2) \neq 1$ und $\mathrm{ggt}(a_1, a_2) \notin R^*$ ergeben sich weitere Lösungen.

5.5.2 Spezielle homogene lineare Gleichungssysteme über Ringen

In diesem Abschnitt werden wir zwei homogene lineare Gleichungssysteme betrachten, die bei der Untersuchung des Konjugationsproblems und des speziellen Konjugationsproblems in den beiden folgenden Abschnitten auftreten werden.

Sei zunächst das folgende lineare Gleichungssystem gegeben:

$$\begin{array}{rcrcrcl} a_{11} \cdot x_1 & & + & a_{13} \cdot x_3 & + & a_{14} \cdot x_4 & = 0 \\ & a_{22} \cdot x_2 & + & a_{23} \cdot x_3 & + & a_{24} \cdot x_4 & = 0 \end{array}$$

Mittels Satz 24 erhält man folgende Lösungen für die erste Gleichung:
$x_1 = r_1 \cdot a_{13} + s_1 \cdot a_{14}$,
$x_3 = -r_1 \cdot a_{11} + t_1 \cdot a_{14}$ und
$x_4 = -s_1 \cdot a_{11} - t_1 \cdot a_{13}$ für $r_1, s_1, t_1 \in R$.

Und analog folgende Lösungen für die 2. Gleichung:
$x_2 = r_2 \cdot a_{23} + s_2 \cdot a_{24}$,
$x_3 = -r_2 \cdot a_{22} + t_2 \cdot a_{24}$ und
$x_4 = -s_2 \cdot a_{22} - t_2 \cdot a_{23}$ für $r_2, s_2, t_2 \in R$.

Die gesuchte Lösung muss beide Gleichungen erfüllen, d.h. die $r_1, r_2, s_1, s_2, t_1, t_2 \in R$ müssen so gewählt werden, dass die beiden Gleichungen $r_1 a_{11} + t_1 a_{14} = r_2 a_{22} + t_2 a_{24}$ und $s_1 a_{11} - t_1 a_{13} = s_2 a_{22} - t_2 a_{23}$ gelten.

Dies ist erfüllt, wenn $r_1 = a_{14} \cdot (m - a_{22} a_{24} k)$, $r_2 = a_{24} \cdot (n - a_{11} a_{14} k)$, $s_1 = a_{13} \cdot (a_{22} a_{23} l - m)$, $s_2 = a_{23} \cdot (a_{11} a_{13} l - n)$, $t_1 = a_{11} \cdot m$ und $t_2 = a_{22} \cdot n$ für beliebige $k, l, m, n \in R$ gewählt werden.

Insgesamt erhält man $x_1 = a_{13} a_{14} \cdot (a_{22} a_{23} l - a_{22} a_{24} k)$, $x_2 = a_{23} a_{24} \cdot (a_{11} a_{13} l - a_{11} a_{14} k)$, $x_3 = a_{22} a_{24} a_{11} a_{14} k$ und $x_4 = -a_{22} a_{23} a_{11} a_{13} l$ für beliebige $k, l \in R$ als Lösungen des gegebenen Gleichungssystems.

Wir suchen nun nach Lösungen für Gleichungssysteme der Form:

$$\begin{array}{rcrcrcl} a_{11} \cdot x_1 & + & & & & a_{14} \cdot x_4 & = 0 \\ & & a_{22} \cdot x_2 & + & & a_{24} \cdot x_4 & = 0 \\ & & & & a_{33} \cdot x_3 & + \; a_{34} \cdot x_4 & = 0 \end{array}$$

Satz 24 liefert für die einzelnen Gleichungen die folgenden Lösungen:
$x_1 = a_{14} \cdot r$ und $x_4 = -a_{11} \cdot r$ für $r \in R$,
$x_2 = a_{24} \cdot s$ und $x_4 = -a_{22} \cdot s$ für $s \in R$ und
$x_3 = a_{34} \cdot t$ und $x_4 = -a_{33} \cdot t$ für $t \in R$.

Um die Lösungen zu bestimmen, die alle drei Gleichungen erfüllen, suchen wir $r, s, t \in R$ mit $a_{11} r = a_{22} s = a_{33} t$. Diese Gleichungskette ist erfüllt für $r = a_{22} a_{33} k$, $s = a_{11} a_{33} k$ und $t = a_{11} a_{22} k$ mit $k \in R$.

Das angegebene Gleichungssystem hat also (mindestens) folgende Lösungen:
$x_1 = -a_{22} a_{33} a_{14} k$, $x_2 = -a_{11} a_{33} a_{24} k$, $x_3 = -a_{11} a_{22} a_{34} k$ und $x_4 = a_{11} a_{22} a_{33} k$ für $k \in R$.

5.5.3 Das Konjugationsproblem in GL(2,R)

Seien $X, Y, Z \in GL(2,R)$ mit $Y = Z \cdot X \cdot Z^{-1}$. Dann ist X, Y eine Instanz des Konjugationsproblems in $GL(2,R)$. Um dieses zu lösen, betrachten wir die zu $Y = Z \cdot X \cdot Z^{-1}$ äquivalente Gleichung $Y \cdot Z = Z \cdot X$.

Für $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$, $Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}$ und $Z = \begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix}$ ergibt sich für $y_3 \neq 0$ das folgende lineare Gleichungssystem (siehe Abschnitt 2.7.4):

$$\begin{array}{rcrcrcll} -y_3 \cdot z_1 & & & + & (x_1 - y_4) \cdot z_3 & + & x_3 \cdot z_4 & = 0 \quad (I) \\ & - & y_3 \cdot z_2 & + & x_2 \cdot z_3 & + & (x_4 - y_4) \cdot z_4 & = 0 \quad (II) \end{array}$$

Mit den Überlegungen aus Abschnitt 5.5.2 erhält man sofort die folgenden Lösungen dieses Gleichungssystems:

$$\begin{aligned} z_1 &= (x_1 - y_4)x_3(y_3(x_4 - y_4)k - x_2y_3l), \\ z_2 &= (x_4 - y_4)x_2(y_3x_3k - y_3(x_1 - y_4)l), \\ z_3 &= (y_3)^2x_3(x_4 - y_4)k, \\ z_4 &= -(y_3)^2x_2(x_1 - y_4)l, \end{aligned}$$

wobei $k, l \in R$. Wählt man k und l derart, dass die resultierende Matrix Z invertierbar ist, so ist die Lösung des Gleichungssystems auch eine Lösung der Instanz X, Y des Konjugationsproblems in $GL(2,R)$.

5.5.4 Das spezielle Konjugationsproblem in GL(2,R)

Sei eine Abbildung $Inn(g) : GL(2,R) \to GL(2,R)$ gegeben. Wir nehmen an, dass $Inn(g)$ als Black Box gegeben ist, d.h. ein Angreifer kann sich zwar Funktionswerte berechnen lassen, hat aber keinerlei Informationen über das benutzte g. Auf diese Art kann man zeigen, dass die angestellten Berechnungen unabhängig von der für $Inn(g)$ gewählten Darstellung sind. Wir zeigen nun, dass in $GL(2,R)$ das spezielle Konjugationsproblem effizient durch die Lösung simultaner Instanzen des Konjugationsproblems lösbar ist.

Seien $B, C, X \in GL(2,R)$. Dann bilden die Werte $B, XBX^{-1} = \widehat{B} = \begin{pmatrix} \widehat{b}_1 & \widehat{b}_2 \\ \widehat{b}_3 & \widehat{b}_4 \end{pmatrix}$ und $C, XCX^{-1} = \widehat{C} = \begin{pmatrix} \widehat{c}_1 & \widehat{c}_2 \\ \widehat{c}_3 & \widehat{c}_4 \end{pmatrix}$ zwei simultane Instanzen des Konjugationsproblems in $GL(2,R)$.

Sei ferner $\widehat{X} \in GL(2,R)$ eine Lösung dieser beiden Instanzen. Dann lässt sich $\widehat{X}$ darstellen als $\widehat{X} = Z \cdot X$ für ein $\begin{pmatrix} z_1 & z_2 \\ z_3 & z_4 \end{pmatrix} = Z \in Z(\widehat{B}) \cap Z(\widehat{C})$. Durch einen Vergleich der

Komponenten von $Z \cdot \widehat{B}$ und $\widehat{B} \cdot Z$ und der Komponenten von $Z \cdot \widehat{C}$ und $\widehat{C} \cdot Z$ erhält man:[15]

- $z_2(\widehat{c}_3\widehat{b}_2 - \widehat{b}_3\widehat{c}_2) = 0$ und $z_3(\widehat{c}_3\widehat{b}_2 - \widehat{b}_3\widehat{c}_2) = 0$
- $z_2(\widehat{c}_2(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_2(\widehat{c}_1 - \widehat{c}_4)) = 0$ und $z_3(\widehat{c}_2(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_2(\widehat{c}_1 - \widehat{c}_4)) = 0$
- $z_2(\widehat{c}_3(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_3(\widehat{c}_1 - \widehat{c}_4)) = 0$ und $z_3(\widehat{c}_3(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_3(\widehat{c}_1 - \widehat{c}_4)) = 0$

Wenn $\widehat{c}_3\widehat{b}_2 = \widehat{b}_3\widehat{c}_2$, $\widehat{c}_2(\widehat{b}_1 - \widehat{b}_4) = \widehat{b}_2(\widehat{c}_1 - \widehat{c}_4)$ und $\widehat{c}_3(\widehat{b}_1 - \widehat{b}_4) = \widehat{b}_3(\widehat{c}_1 - \widehat{c}_4)$ erfüllt sind, dann gilt $\widehat{B} \in Z(\widehat{C})$. Wurden $\widehat{B}, \widehat{C} \in GL(2,R)$ so gewählt, dass $\widehat{B} \notin Z(\widehat{C})$, so ist eine dieser Gleichungen nicht erfüllt und z_2 und z_3 müssen Nullteiler sein.

Wurden $\widehat{B}, \widehat{C} \in GL(2,R)$ so gewählt, das einer der Werte $\widehat{c}_3\widehat{b}_2 - \widehat{b}_3\widehat{c}_2$, $\widehat{c}_2(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_2(\widehat{c}_1 - \widehat{c}_4)$ oder $\widehat{c}_3(\widehat{b}_1 - \widehat{b}_4) - \widehat{b}_3(\widehat{c}_1 - \widehat{c}_4)$ kein Nullteiler ist, so folgt ferner, dass $z_2 = z_3 = 0$ gilt. Ist eines der Ringelemente $\widehat{b}_2$, $\widehat{b}_3$, $\widehat{c}_2$ oder $\widehat{c}_3$ kein Nullteiler, so folgt, dass Z von der Form $Z = \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}$ für ein $z_1 \in R$ ist. Da für solche Z die Gleichung $Z \cdot M = M \cdot Z$ für alle $M \in M(2,R)$ erfüllt ist, liegt Z im Zentrum von $M(2,R)$ und damit insbesondere im Zentrum von $GL(2,R)$. Somit gilt $Inn(X) = Inn(\widehat{X})$, d.h. $\widehat{X} \in GL(2,R)$ ist eine Lösung der Instanz $Inn(X)$ des speziellen Konjugationsproblems in $GL(2,R)$.

Wir zeigen nun, dass simultane Lösungen zweier Instanzen des Konjugationsproblems effizient berechnet werden können. Die Gleichungen $XBX^{-1} = \widehat{B}$ und $XCX^{-1} = \widehat{C}$ sind äquivalent zu den Gleichungen $XB = \widehat{B}X$ und $XC = \widehat{C}X$. Wenn $B \notin Z(C)$ gilt, so erhält man ein System von 3 linearen Gleichungen. Im Angriff aus Abschnitt 5.3 können die Elemente $\widehat{B}, \widehat{C} \in GL(2,R)$ beliebig gewählt werden. Ist $\widehat{b}_3$ invertierbar, so ist das erhaltene Gleichungssystem äquivalent zu:

$$\begin{array}{rcrcrcl} x_1 & + & & & \frac{\widehat{b}_4 - b_1}{\widehat{b}_3} \cdot x_3 & - & \frac{b_3}{\widehat{b}_3} \cdot x_4 = 0 \\ & & x_2 & - & \frac{b_2}{\widehat{b}_3} \cdot x_3 & + & \frac{\widehat{b}_4 - b_4}{\widehat{b}_3} \cdot x_4 = 0 \\ & & & & (\widehat{c}_4 - c_1 - \widehat{c}_3 \cdot \frac{\widehat{b}_4 - b_1}{\widehat{b}_3}) \cdot x_3 & - & (c_3 - \widehat{c}_3 \cdot \frac{b_3}{\widehat{b}_3}) \cdot x_4 = 0 \end{array}$$

Für beliebige $r \in R$ ist $x_1 = k_1 \cdot r$, $x_2 = k_2 \cdot r$, $x_3 = k_3 \cdot r$ und $x_4 = k_4 \cdot r$ eine Lösung dieses Systems, wobei $k_4 = \widehat{c}_4 - c_1 - \widehat{c}_3 \cdot \frac{\widehat{b}_4 - b_1}{\widehat{b}_3}$, $k_3 = (c_3 - \widehat{c}_3 \frac{b_3}{\widehat{b}_3}) \cdot k_4$, $k_2 = \frac{b_2}{\widehat{b}_3} \cdot k_3 - \frac{\widehat{b}_4 - b_4}{\widehat{b}_3} \cdot k_4$ und $k_1 = \frac{b_3}{\widehat{b}_3} \cdot k_4 - \frac{\widehat{b}_4 - b_1}{\widehat{b}_3} \cdot k_3$.

Ist $\widehat{b}_3 c_3 - \widehat{c}_3 b_3$ oder $\widehat{b}_3(\widehat{c}_4 - c_1) - \widehat{c}_3(\widehat{b}_4 - b_1)$ kein Nullteiler, so ist $\begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \end{pmatrix} \in GL(2,R)$ und es gilt $\begin{pmatrix} rk_1 & rk_2 \\ rk_3 & rk_4 \end{pmatrix} \neq \begin{pmatrix} sk_1 & sk_2 \\ sk_3 & sk_4 \end{pmatrix}$ für $r, s \in R$ mit $r \neq s$, d.h. der obige Ansatz liefert uns $|R|$ verschiedene Lösungen (aus $M(2,R)$) der Gleichungen $XB = \widehat{B}X$ und $XC = \widehat{C}X$. In

[15] Da $\widehat{X}$ auch durch $\widehat{X} = X \cdot \widehat{Z}$ für ein $\widehat{Z} \in Z(B) \cap Z(C)$ dargestellt werden kann, bleibt der folgende Absatz richtig, wenn $\widehat{b}_i$ und $\widehat{c}_i$ durch b_i und c_i ersetzt werden. Insbesondere gilt $B \in Z(C) \Leftrightarrow \widehat{B} \in Z(\widehat{C})$.

diesem Fall wissen wir ferner, dass $\begin{pmatrix} rk_1 & rk_2 \\ rk_3 & rk_4 \end{pmatrix} \in GL(2,R)$ genau dann gilt, wenn $r \in R$ invertierbar ist.
Da $X \in GL(2,R)$ invertierbar ist, sind die Gleichungen $XB = \widehat{B}X$ und $\widehat{B} = XBX^{-1}$ äquivalent. Für ein Element $\widehat{X} \in M(2,R)$ mit $\widehat{X}B = \widehat{B}\widehat{X}$ folgt $(X^{-1}\widehat{X})B = B(X^{-1}\widehat{X})$, d.h. $\widehat{X} = X \cdot Z$ mit $Z \in Z_{M(2,R)}(B)$.
Also haben die simultanen Lösungen der Gleichungen $XB = \widehat{B}X$ und $XC = \widehat{C}X$ (in $M(2,R)$) die Form $Z \cdot X$ wobei $Z \in Z_{M(2,R)}(\hat{B}) \cap Z_{M(2,R)}(\widehat{C})$ ist. Wurden die Elemente $\widehat{B}, \widehat{C} \in GL(2,R)$ so gewählt, dass $Z_{M(2,R)}(\widehat{B}) \cap Z_{M(2,R)}(\widehat{C}) = Z(M(2,R))$ gilt, dann gibt es genau $|Z(M(2,R))| = |R|$ solcher simultaner Lösungen, d.h. über die Darstellung $x_1 = k_1 \cdot r$, $x_2 = k_2 \cdot r$, $x_3 = k_3 \cdot r$ und $x_4 = k_4 \cdot r$ mit $r \in R$ haben wir bereits alle Lösungen gefunden.

Für die Matrizen $B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ und $C = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ sind obige Bedingungen erfüllt, d.h. es gilt $Z(B) \cap Z(C) = Z(GL(2,R))$ und die entsprechenden Werte sind keine Nullteiler. Für simultane Lösungen $\widehat{X}$ der beiden Instanzen $B, \widehat{B} = XBX^{-1}$ und $C, \widehat{C} = XCX^{-1}$ gilt also $Inn(X) = Inn(\widehat{X})$. Um für einen gegebenen inneren Automorphismus $Inn(g)$ das spezielle Konjugationsproblem zu lösen, kann man also wie folgt vorgehen: Man berechnet die Bilder von $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ und $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ unter $Inn(g)$ und löst anschließend dann die resultierenden simultanen Instanzen des Konjugationsproblems in $GL(2,R)$.

6 MOR auf p-Gruppen

Dieses Kapitel beschäftigt sich mit der Frage, inwieweit sich p-Gruppen für einen Einsatz im MOR System eignen. Eine (nicht-abelsche) Gruppe G muss die folgenden (notwendigen) Bedingungen erfüllen, um im MOR System einsetzbar zu sein (siehe auch Abschnitt 3.1.1):

1. Das Zentrum $Z(G)$ von G darf
   - nicht zu klein sein (ansonsten kann das DLP in $< Inn(G) >$ auf das DLP und das SCP in G reduziert werden) und
   - nicht zu groß sein (ansonsten ist die Gruppe $Inn(G)$ zu klein).
2. Es sollte eine Darstellung für die Gruppenelemente geben, für die effiziente Methoden
   - zur Speicherung der Elemente in einem Computer und
   - zur Durchführung der Gruppenoperation auf einem Computer

   exitieren.
3. Die Gruppe G muss endlich erzeugt sein und in G muss das Wortproblem effizient lösbar sein (da die Abbildungen $Inn(g)$, $Inn(g^a)$ durch die Bilder der Generatoren der Gruppe G gegeben sind).
4. Das DLP muss schwer sein in $Inn(G)$ (ansonsten kann der geheime Schlüssel a berechnet werden).

Beim Einsatz von direkten oder semi-direkten Produkten vererben sich viele Eigenschaften der Faktoren G_1 und G_2 auf ihr Produkt $G_1 \times G_2$ bzw. $G_1 \times_\theta G_2$:
Sei $G = G_1 \times G_2$ bzw. $G = G_1 \times_\theta G_2$ für einen Homomorphismus $\theta : G_2 \to Aut(G_1)$ das direkte bzw. semi-direkte Produkt der Gruppen G_1 und G_2. Seien ferner $g_{11}, \ldots, g_{1n}$ und $g_{21}, \ldots, g_{2m}$ Generatoren und e_1 und e_2 die neutralen Elemente von G_1 bzw. G_2. Dann wird die Gruppe G erzeugt von $(g_{11}, e_2), \ldots (g_{1n}, e_2), (e_1, g_{21}), \ldots, (e_1, g_{2m})$. Mit G_1 und G_2 ist also auch G endlich erzeugt. Ist das Wortproblem in G_1 und G_2 effizient lösbar, so gilt dies auch für G.
Im Falle eines direkten Produktes ist G effizient implementierbar, wenn dies für die Faktoren G_1 und G_2 gilt. Bei semi-direkten Produkten muss zusätzlich gefordert werden, dass die Bilder $\theta(g_2)(g_1)$ für beliebige $g_1 \in G_1$ und $g_1 \in G_2$ effizient berechnet werden können.
Für direkte Produkte gilt, dass $|Z(G)| = |Z(G_1)| \cdot |Z(G_2)|$. Im Falle von semi-direkten Produkten gilt $|Z(G_1)| \leq |Z(G)| \leq |Z(G_1)| \cdot |Z(G_2)|$. In beiden Fällen lässt sich also durch geeignete Wahl von G_1 und G_2 Einfluss auf die Größe des Zentrums nehmen.
Für die Schwierigkeit der Berechnung von diskreten Logarithmen gelten die Reduktionen $DLP_{G_1} \leq_P DLP_G$ und $DLP_{G_2} \leq_P DLP_G$, da sich Lösungen der Instanzen g, g^a und h, h^b des DLP in G_1 bzw. G_2 durch Lösen der Instanzen $(g, e_2), (g^a, e_2) = (g, e_2)^a$ und $(e_1, h), (e_1, h^b) = (e_1, h)^b$ des DLP in G berechnen lassen. Zusammen mit der Reduktion $DLP_G \leq_P DLG_{Inn(G)}$ (siehe Satz 13 auf Seite 41) folgt also, dass $DLP_{G_1} \leq_P DLG_{Inn(G)}$

und $DLP_{G_2} \leq_P DLG_{Inn(G)}$ gelten. Bei direkten Produkten ist das DLP in G effizient lösbar, wenn es in einem der Faktoren G_1 oder G_2 effizient lösbar ist. Auch für semi-direkte Produkte besteht die Gefahr, dass die Schwierigkeit des DLP in G auf die Schwierigkeit des DLP in den Faktoren G_1 oder G_2 reduziert werden kann. Ein Beispiel dafür ist die in Kapitel 4 betrachtete Gruppe $G = SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$. Für die Analyse von MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ haben wir ausgenutzt, dass für $g = (x, y) \in G$ aus $Inn(g)$ und $Inn(g^a)$ die Werte $\pm(x\theta_1(y))$ und $\pm(x\theta_1(y))^a$ berechnet werden können und damit $DLP_{SL(2,\mathbb{Z}_p)} \geq_P DLP_G$ gilt.

Ein vielversprechender Ansatz scheint die Betrachtung von p-Gruppen zu sein. Im weiteren Verlauf dieses Abschnittes werden wir untersuchen, inwiefern p-Gruppen die oben definierten Anforderungen erfüllen und für eine Benutzung im MOR System geeignet sind.

Größe des Zentrums: Für die Benutzung einer Gruppe G im MOR System fordern wir, dass sowohl $|Z(G)|$ als auch $|G/Z(G)|$ exponentiell (im Sicherheitsparameter $\log p$) sind. Die Größe des Zentrums von p-Gruppen lässt sich anhand der folgenden Sätze abschätzen.

Satz 25 *Sei $G \neq \{e\}$ eine endliche p-Gruppe. Dann hat G ein nicht-triviales Zentrum $Z \neq \{e\}$.*

Beweis: Siehe [58].

Satz 26 *Sei G eine nicht-abelsche p-Gruppe. Dann ist p^2 ein Teiler von $|G/Z(G)|$.*

Beweis: Siehe [21].

Korollar 4 *Sei $G \neq \{e\}$ eine endliche, nicht-abelsche p-Gruppe der Ordnung p^r. Dann gilt*

$$p \leq |Z(G)| \leq p^{r-2}.$$

Damit erfüllen nicht-abelsche p-Gruppen die Anforderungen an die Größe des Zentrums, wenn für p eine hinreichend große Primzahl gewählt wird.

Effiziente Implementierbarkeit und Wortproblem: In [26] wird ein allgemeines Konstruktionsprinzip zur effizienten Implementierung von p-Gruppen gegeben:

Sei G eine endliche p-Gruppe der Ordnung p^n. Dann lässt sich für G eine Normalreihe

$$G = G_0 \triangleright G_1 \triangleright \cdots \triangleright G_n = \{e\}$$

angeben, deren Faktoren G_{i-1}/G_i für $1 \leq i \leq n$ Ordnung p haben und damit zyklisch sind. Sei $\{e\} \neq g_i \in G_{i-1}/G_i$, so ist g_i Generator von G_{i-1}/G_i. Die Gruppe G_{i-1} wird erzeugt von G_i und g_i und die Elemente $g_1, \ldots, g_n$ erzeugen ganz G.

Die Gruppe G kann als $G = <g_1, \ldots, g_n \mid R>$ mit den folgenden Relationen R dargestellt werden:

$$\begin{aligned} g_i^p &= \prod_{k=i+1}^{n} g_k^{x_{i,k}} \text{ für } 1 \leq i \leq n \quad \text{und} \\ [g_i, g_j] &= \prod_{k=i+1}^{n} g_k^{x_{i,j,k}} \text{ für } 1 \leq j < i \leq n, \end{aligned}$$

wobei alle Exponenten $x_{i,k}$ und $x_{i,j,k}$ zwischen 0 und $p-1$ liegen. Eine solche Darstellung einer Gruppe heißt *power-commutator presentation* (oder kurz PCP).[16]

Jedes Gruppenelement $g \in G$ kann eindeutig in der Form

$$g = g_1^{x_1} \cdots g_n^{x_n} \text{ mit } 0 \leq x_i < p \text{ für } 1 \leq i \leq n$$

geschrieben werden, d.h. die Abbildung $exp : (\mathbb{Z}_p)^n \to G, (x_1, \cdots, x_n) \mapsto g_1^{x_1} \cdots g_n^{x_n}$ ist eine Bijektion. Eine Darstellung eines Gruppenelementes g als $g = g_1^{x_1} \ldots g_n^{x_n}$ mit $0 \leq x_i < p$ für $1 \leq i \leq n$ nennen wir Normalform von g (bzgl. $g_1, \ldots, g_n$). Die Elemente aus G können als Exponenten-Tupel $(x_1, \ldots, x_n)$ effizient im Computer gespeichert werden.

Im MOR System sind die inneren Automorphismen $Inn(g)$ durch die Bilder $Inn(g)(g_i)$ der Generatoren g_i von G gegeben. Um Bilder unter $Inn(g)$ berechnen zu können, muss es einen effizienten Algorithmus geben, der ein gegebenes Gruppenelement als Produkt der Generatoren darstellt. Dieses Problem entfällt bei Speicherung der Gruppenelemente als Exponenten-Tupel.

Wir beschäftigen uns nun mit der Frage, wie Verknüpfungen in G sich effizient berechnen lassen. Seien die Elemente $g = g_1^{x_1} \cdots g_n^{x_n}$ und $h = g_1^{y_1} \cdots g_n^{y_n}$ gegeben. Dann gilt $g \cdot h = g_1^{x_1} \cdots g_n^{x_n} g_1^{y_1} \cdots g_n^{y_n}$. Um dieses Element wieder in Normalform zu bringen, wendet man die angegebenen Relationen an. Das dabei verwendete Vorgehen hängt stark von den gegebenen Relationen ab. Einen Überblick über effiziente Verfahren zur Normalisierung liefert [18].

Schwierigkeit des DLP in Inn(G): In Satz 13 auf Seite 41 wurde bereits gezeigt, dass die Berechnung von diskreten Logarithmen in $<Inn(g)>$ bei geeigneter Wahl von $g \in G$ immer mindestens so schwer ist, wie die Berechnung von diskreten Logarithmen in $<g>$. Ein Ansatz zu gewährleisten, dass die Berechnung von diskreten Logarithmen schwer ist in $Inn(G)$ besteht in der Wahl einer Gruppe G, in der diskrete Logarithmen nicht effizient berechnet werden können. Dieser Ansatz funktioniert bei der Darstellung einer Gruppe als PCP leider nicht, wie der folgende Satz zeigt.

[16] Ein anderer gebräuchlicher Name ist PAG-System.

Satz 27 *Sei $G = <x_1, \ldots, x_n \mid R>$ eine Gruppe, die als PCP gegeben ist. Dann lassen sich diskrete Logarithmen in G effizient berechnen.*

Beweis: Seien mit $g = g_1^{x_1} \cdots g_n^{x_n}$ und $h = g_1^{y_1} \cdots g_n^{y_n}$ zwei Elemente aus G gegeben. Durch die Relationen $[g_i, g_1] = \prod_{k=i+1}^{n} g_k^{x_{i,j,k}}$ für $1 < i \leq n$ hat die Normalform des Produktes $g \cdot h = g_1^{x_1} \cdots g_n^{x_n} g_1^{y_1} \cdots g_n^{y_n}$ die Form $g \cdot h = g_1^{x_1+y_1 \bmod p} g_2^{z_2} \cdots g_n^{z_n}$ mit $0 \leq z_i < p$ für $2 \leq i \leq n$.
Die Normalform von g^a hat also die Form $g^a = g_1^{a \cdot x_1 \bmod p} g_2^{z_2} \cdots g_n^{z_n}$ mit $0 \leq z_i < p$ für $2 \leq i \leq n$. Aus den Werten x_1 und $a \cdot x_1 \bmod p$ kann der gesuchte diskrete Logarithmus a leicht berechnet werden. □

Eine notwendige Bedingung für die Schwierigkeit des DLP in $<Inn(g)>$ ist, dass die Ordnung von $<Inn(g)>$ so groß gewählt wurde, dass die Laufzeit von generischen Algorithmen wie Baby-Step-Giant-Step oder Pollard's Rho Methode (siehe auch Abschnitt 2.3 auf Seite 15) zu hoch ist, um in $<Inn(g)>$ (in der Praxis) diskrete Logarithmen berechnen zu können.
Sei G eine nicht-abelsche p-Gruppe der Ordnung p^n und sei $g \in G$. Dann hat $Inn(g)$ die Ordnung p^r für ein $r < n-1$. Der Exponent r geht dabei nur linear in die Laufzeit bei der Berechnung diskreter Logarithmen ein:
Sei mit $Inn(g), Inn(g^x)$ eine Instanz des DLP in $<Inn(g)>$ gegeben. Durch Berechnung von $(Inn(g))^{p^{r-1}}$ und $(Inn(g^x))^{p^{r-1}}$ erhält man eine Instanz des DLP in der von $(Inn(g))^{p^{r-1}}$ erzeugten Untergruppe der Ordnung p. In dieser Gruppe können diskrete Logarithmen mittels generischer Algorithmen in einer Zeit von $\mathcal{O}(\sqrt{p})$ berechnet werden. Als Ergebnis erhält man $x_1 = x \bmod p$, d.h. $x = l_1 \cdot p + x_1$ für ein $l_1 \in \mathbb{N}$ mit $l_1 < p^{r-1}$.
Als nächstes betrachten wir die Instanz $g^p, g^{x-x_1} = (g^p)^{l_1}$ des DLP. Analog zum vorigen Schritt kann $x_2 = l_1 \bmod p$ berechnet werden, d.h. $l_1 = l_2 \cdot p + x_2$ für ein $l_2 \in \mathbb{N}$ mit $l_2 < p^{r-2}$. Dieses Verfahren r-fach iteriert liefert $l_r \bmod p = l_r$. Über die Formel $x_i = l_{i+1} \cdot p + x_{i+1}$ lässt sich $x = x_0$ berechnen.
Diskrete Logarithmen in $<Inn(g)>$ lassen sich also in Zeit $\mathcal{O}(k \cdot \sqrt{p})$ berechnen. Der Exponent k hat dabei nur linearen Einfluss auf die Laufzeit, die Größe $\log p$ der benutzten Primzahl jedoch exponentiellen. Um zu verhindern, dass mittels generischer Algorithmen diskrete Logarithmen in p-Gruppen der Ordnung p^n berechnet werden können, sollte p in der Größenordnung 160 Bits gewählt werden.

6.1 Beispiel: (nicht-abelsche) p-Gruppen der Ordnung p^3

Die im letzten Abschnitt gemachten Aussagen werden am Beispiel von (nicht-abelschen) p-Gruppen der Ordnung p^3 noch einmal verdeutlicht. Für nicht-abelsche p-Gruppen der Ordnung p^3 existieren genau die beiden folgenden Isomorphietypen:

- $G = <x_1, x_2, x_3 \mid x_1^p = x_2^p = x_3^p = 1, [x_3, x_2] = [x_3, x_1] = 1, [x_2 x_1] = x_3>$
- $G = <x_1, x_2 \mid x_2^{p^2} = x_1^p = 1, x_1^{-1} x_2 x_1 = x_2^{p+1}>$
  $= <x_1, x_2, x_3 \mid x_1^p = 1, x_2^p = x_3, x_3^p = 1, [x_2, x_1] = x_3, [x_3, x_1] = 1, [x_3, x_2] = 1>$

In beiden Fällen hat das Zentrum $Z(G)$ die Größe p und wird von x_3 erzeugt. Im zweiten Fall handelt es sich um die metazyklische (nicht-abelsche) Gruppe der Ordnung p^3.

1. Beispiel: $G = < x_1, x_2, x_3 \mid x_1^p = x_2^p = x_3^p = 1, [x_3, x_2] = [x_3, x_1] = 1, [x_2 x_1] = x_3 >$

Aus $[x_2 x_1] = x_3$ folgt, dass $x_2^m x_1^n = x_1^n x_2^m x_3^{nm}$ gilt. Für $g = x_1^{a_1} x_2^{a_2} x_3^{a_3}$ und $h = x_1^{b_1} x_2^{b_2} x_3^{b_3}$ ergibt sich

$$\begin{aligned} g \cdot h &= x_1^{a_1} x_2^{a_2} x_3^{a_3} x_1^{b_1} x_2^{b_2} x_3^{b_3} \\ &= x_1^{a_1} x_2^{a_2} x_1^{b_1} x_2^{b_2} x_3^{a_3+b_3} \\ &= x_1^{a_1+b_1} x_2^{a_2+b_2} x_3^{a_3+b_3+a_2 \cdot b_1}. \end{aligned}$$

Sind g und h durch ihre Exponenten-Tupel (a_1, a_2, a_3) bzw. (b_1, b_2, b_3) gegeben, so entspricht dem Produkt $g \cdot h$ das Tupel $(a_1 + b_1 \bmod p, a_2 + b_2 \bmod p, a_3 + b_3 + a_2 \cdot b_1 \bmod p)$. Produkte in G können also durch 3 Additionen und eine Multiplikation in $\mathbb{Z}_p$ berechnet werden. Das Inverse von g ergibt sich als $g^{-1} = x_3^{-a_3} x_2^{-a_2} x_1^{-a_1} = x_1^{-a_1} x_2^{-a_2} x_3^{-a_3+a_1 a_2}$ und Konjugationen sind von der Form

$$\begin{aligned} g \cdot h \cdot g^{-1} &= (x_1^{a_1+b_1} x_2^{a_2+b_2} x_3^{a_3+b_3+a_2 \cdot b_1}) \cdot (x_1^{-a_1} x_2^{-a_2} x_3^{-a_3+a_1 a_2}) \\ &= x_1^{b_1} x_2^{b_2} x_3^{b_3+a_2 b_1 - a_1 b_2}. \end{aligned}$$

Das Konjugationsproblem ist leicht in G: Sei mit $h = x_1^{b_1} x_2^{b_2} x_3^{b_3}$ und $ghg^{-1} = x_1^{c_1} x_2^{c_2} x_3^{c_3}$ eine Instanz des Konjugationsproblems in G gegeben. Dann ist die Lösungsmenge dieser Instanz gegeben durch $L = \{g = x_1^{a_1} x_2^{a_2} x_3^{a_3} \in G \mid b_3 + a_2 b_1 - a_1 b_2 = c_3, a_3 \in \mathbb{Z}_p\}$.
Das SCP kann in G durch Lösen zweier simultaner Instanzen $h_1 = g_1^{b_{11}} g_2^{b_{21}} g_3^{b_{31}}, gh_1 g^{-1}$ und $h_2 = g_1^{b_{12}} g_2^{b_{22}} g_3^{b_{32}}, gh_2 g^{-1}$ gelöst werden, für die die beiden Exponenten-Vektoren (b_{11}, b_{21}) und (b_{12}, b_{22}) linear unabhängig sind. Insbesondere können also bei gegebenem $Inn(g)$ die beiden ersten Exponenten a_1 und a_2 der Normalform von g berechnet werden.
Ist mit $Inn(g)$ und $Inn(g^a)$ eine Instanz des DLP in $< Inn(g) >$ gegeben, so können die jeweils ersten Exponenten a_1 und $a \cdot a_1 \bmod p$ der Normalformen von g bzw. g^a berechnet werden. Das diskrete Logarithmus Problem in $Inn(G)$ kann auf diese Weise effizient gelöst werden. Diese Gruppe ist also (mit der vorgestellten Implementierung als PCP) nicht für das MOR System geeignet.

2. Beispiel: $G = < x_1, x_2, x_3 \mid x_1^p = x_3^p = 1, x_2^p = x_3, [x_2, x_1] = x_3, [x_3, x_1] = [x_3, x_2] = 1 >$.

Analog zum ersten Beispiel ergibt sich für $g = x_1^{a_1} x_2^{a_2} x_3^{a_3}$ und $g = x_1^{b_1} x_2^{b_2} x_3^{b_3}$ das Produkt $g \cdot h$ als:

$$\begin{aligned} g \cdot h &= x_1^{a_1} x_2^{a_2} x_3^{a_3} x_1^{b_1} x_2^{b_2} x_3^{b_3} \\ &= x_1^{a_1} x_2^{a_2} x_1^{b_1} x_2^{b_2} x_3^{a_3+b_3} \\ &= x_1^{a_1+b_1} x_2^{a_2+b_2} x_3^{a_3+b_3+a_2 \cdot b_1} \\ &= x_1^{(a_1+b_1 \bmod p)} x_2^{(a_2+b_2 \bmod p)} x_3^{(a_3+b_3+a_2 \cdot b_1 + \frac{a_2+b_2-(a_2+b_2 \bmod p)}{p} \bmod p)}. \end{aligned}$$

Der zusätzliche Term $\frac{a_2+b_2-(a_2+b_2 \bmod p)}{p}$ im Exponenten von x_3 kommt zustande, da $x_2^p = x_3$ und nicht wie im ersten Beispiel $x_2^p = 1$ gilt. Er hat den Wert 1, wenn $b_2 + b_3 \geq p$ gilt, und ansonsten den Wert 0. Bei der Konjugation fällt dieser Term allerdings wieder weg, so dass genau wie im ersten Beispiel

$$g \cdot h \cdot g^{-1} \quad = \quad x_1^{b_1} x_2^{b_2} x_3^{b_3+a_2b_1-a_1b_2}$$

gilt. Analog zum ersten Beispiel können auch in dieser Gruppe diskrete Logarithmen in $Inn(G)$ effizient berechnet werden.

6.2 Fazit

Die Anforderungen des MOR Systems bzgl. der Größe des Zentrums, der effizienten Implementierbarkeit und der Lösbarkeit des Wortproblems werden von p-Gruppen in PCP-Darstellung erfüllt. Die Tatsache, dass das DLP in solchen Gruppen leicht ist, erschwert genau wie im Fall von $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ und $GL(2, R) \times_\theta \mathbb{Z}_n$ den Nachweis der Schwierigkeit des DLP in $Inn(G)$ erheblich.
Bei der Darstellung von p-Gruppen als PCP kann man o.B.d.A. immer annehmen, dass $g_1 \notin Z(G)$ gilt. Die Exponenten-Tupel aller Elemente aus $g \cdot Z(G)$ beginnen mit dem selben Wert a_1. Als notwendige Bedingung für die Schwierigkeit des DLP in $< Inn(g) >$ muss gefordert werden, dass für $g = g_1^{a_1} \cdots g_n^{a_n}$ der Exponent a_1 aus $Inn(g)$ nicht effizient berechnet werden kann. Ansonsten können aus $Inn(g)$ und $Inn(g^n)$ die Werte a_1 und $n \cdot a_1 \bmod p$ berechnet und der diskrete Logarithmus n ermittelt werden. Bei der Betrachtung der p-Gruppen der Ordnung p^3 konnte gezeigt werden, dass diese Bedingung nicht erfüllt ist und die Berechnung diskreter Logarithmen in $< Inn(g) >$ effizient möglich ist.

Unter der Annahme, dass für p-Gruppen G der Ordnung p^k mit $k \geq 4$ das DLP in $Inn(G)$ schwer ist, erfüllen diese Gruppen die aufgestellten notwendigen Bedingungen für eine Benutzung im MOR System. Bei der Frage, ob diese Bedingungen hinreichend für die Sicherheit von MOR auf diesen Gruppen sind, handelt es sich derzeit um eine offenes Problem.

7 Kryptografische Probleme und Protokolle

Ziel dieses Kapitels ist die systhematische Untersuchung von kryptografischen Problemen auf nicht-abelschen Gruppen, die durch Kombination von klassischen kryptografischen Problemen, wie dem Problem der Berechnung diskreter Logarithmen oder den Diffie-Hellman-Problemen, mit dem Konjugationsproblem entstehen.

In Abschnitt 7.1 stellen wir zunächst Möglichkeiten vor, wie sich klassische kryptografische Probleme mit dem Konjugationsproblem sinnvoll zu neuen Problemen kombinieren lassen. In der anschließenden Analyse dieser Probleme untersuchen wir, für welche Parameterwahl diese Probleme eine eindeutige Lösung besitzen, welche komplexitätstheoretischen Abhängigkeiten zwischen diesen Problemen bestehen und wie schwer sie in den Matrixgruppen $SL(2, \mathbb{Z}_p)$ und $GL(2, \mathbb{Z}_p)$ sind. Die für die Kryptografie wichtige Frage der Schwierigkeit dieser Probleme untersuchen wir durch Reduktion einzelner Teilprobleme auf das kombinierte Problem. Solche Reduktionen werden für alle vorgestellten Kombinationen durchgeführt. Die kombinierten Probleme sind damit mindestens so schwer wie ihre gut untersuchten Teilprobleme.
Eine ähnliche Konstruktion wie bei den betrachteten Kombinationsproblemen benutzen wir, um in Abschnitt 7.1.3 durch Kombination des DLP und des CP eine Einwegfunktion auf nicht-abelschen Gruppen zu erhalten. Die Einwegfunktion verfügt über Homomorphie- und Blendungseigenschaften, die nützlich für die Konstruktion von kryptografischen Protokollen sind. Mit den definierten Problemen und der Einwegfunktion stehen damit Bausteine zur Verfügung, die zum Design kryptografischer Protokolle auf nicht-abelschen Gruppen eingesetzt werden können.
Einige der vorgestellten Kombinationsprobleme wurden bereits bei der Umsetzung des Diffie-Hellman Protokolls und des ElGamal-Verfahrens auf Zopfgruppen verwendet [23]. Die dabei vorgeschlagenen Protokolle lassen sich auch auf anderen nicht-abelschen Gruppen ausführen. Zunächst stellen wir die allgemeine Version der Protokolle für nicht-abelsche Gruppen vor. Anschließend zeigen wir, wie diese Protokolle auf Zopfgruppen umgesetzt wurden. Die vorgestellten Protokolle lassen sich auch auf der Gruppe $GL(2, \mathbb{Z}_p)$ ausführen. Wir zeigen, dass eine Umsetzung auf $GL(2, \mathbb{Z}_p)$ nicht sicher ist.
Ein weiteres Verfahren, dass auf einer Kombination aus DLP und CP in der Gruppe $GL(2, \mathbb{Z}_n)$ beruht, ist das Cayley-Purser Verschlüsselungsverfahren [16]. Wir zeigen, dass MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ und der Cayley-Purser Algorithmus auf die selbe Weise angreifbar sind. Mit dem selben Angriff kann bei Ausführung der Protokolle aus [23] auf der Gruppe $GL(2, \mathbb{Z}_p)$ der geheime Schlüssel effizient berechnet werden. Wir arbeiten die gemeinsame Konstruktionsschwäche dieser Protkolle heraus, die für das Gelingen des Angriffes verantwortlich ist und geben ein allgemeines Prinzip an, wie diese Schwäche bei der Konstruktion zukünftiger Verfahren vermieden werden kann.
In Abschnitt 7.3 benutzen wir eine Variante der in Abschnitt 7.1 definierten Einwegfunktion, um ein beweisbar sicheres Commitment Scheme auf nicht-abelschen Gruppen zu konstruieren. Anschließend geben wir eine konkrete Realisierung dieses Commitment Schemes auf der Gruppe $GL(2, \mathbb{Z}_p)$ an. Diese Realisierung zeigt, dass mit den in Abschnitt 7.1 ein-

geführten Bausteinen die Konstruktion beweisbar sicherer kryptografischer Verfahren auf nicht-abelschen Gruppen und insbesondere auf der Gruppe $GL(2, \mathbb{Z}_p)$ möglich ist.

7.1 Kryptografische Probleme auf nicht-abelschen Gruppen

In Kapitel 3 wurde die enge Verwandtschaft des DLP in $Inn(G)$ mit dem DLP und dem SCP in G diskutiert. Bei geeigneter Wahl der nicht-abelschen Gruppe G kann das DLP in der Gruppe $Inn(G)$ selbst dann schwer sein, wenn das DLP und das SCP in G effizient lösbar sind.
Kombiniert man das DLP auf analoge Weise mit dem CP, so erhält man das Problem für gegebene Elemente $g, x, g^a x g^{-a}$ den Wert a zu berechnen, das wir Diskreter Logarithmus Konjugationsproblem (DLCP) nennen werden. Das DLCP wird sich als mindestens so schwer wie das DLP herausstellen. Wie im Falle des DLP in $Inn(G)$ kann das DLCP in bestimmten nicht-abelschen Gruppen G sogar selbst dann schwer sein, wenn mit dem DLP und dem CP die beiden Teilprobleme effizient lösbar sind.
In diesem Abschnitt werden wir das CP mit dem DLP und den Diffie-Hellman Problemen kombinieren, um neue Probleme auf nicht-abelschen Gruppen zu erhalten.
Wir geben zunächst eine tabellarische Übersicht über die in diesem Abschnitt betrachteten Probleme, bevor wir diese in den Abschnitten 7.1.1 und 7.1.2 detailliert untersuchen. Ein Schwerpunkt der Betrachtungen bildet dabei die Frage, in welcher Beziehung die neuen Probleme zueinander und zu ihren Teilproblemen stehen.

Im Folgenden nehmen wir an, dass G eine nicht-abelsche Gruppe ist und dass für die Gruppenelemente $x, g \in G$ die Bedingungen $\text{ord}(g)$ prim und $x \notin Z(g)$ erfüllt sind. Dadurch ist sichergestellt, dass viele der neuen Probleme eine eindeutige Lösung besitzen. Tabelle 1 auf der nächsten Seite stellt einen Überblick der im Folgenden betrachteten Probleme dar.

Bemerkung 6 (Namensgebung) *Das CDH-type Konjugationsproblem wurde in der Kryptografie erstmals in [23] für die Verwendung auf Zopfgruppen vorgeschlagen. Die Autoren versäumten es aber, dem Problem einen Namen zu geben. In [9] taucht es als CDH-type Conjugacy Problem, in [29] als Braid DH-Problem und in [10] als Diffie-Hellman Conjugacy Problem (DHCP) auf.*
Da für die ersten drei der in Tabelle 1 definierten Kombinationsprobleme die Verwandtschaft zum diskreten Logarithmus Problem bzw. zu den Diffie-Hellman Problemen offensichtlich ist, bezeichnen wir diese mit DLCP, CDHCP und DDHCP. Für die beiden folgenden Probleme übernehmen wir die Bezeichnungen aus [9], um Verwechslungen zu vermeiden.

Problem	gegeben	gesucht
Klassische Probleme		
Diskreter Logarithmus Problem (DLP)	g, g^a	a
Computational Diffie-Hellman Problem (CDHP)	g, g^a, g^b	g^{ab}
Decisional Diffie-Hellman Problem (DDHP)	g, g^a, g^b, g^r	$r \stackrel{?}{=} ab$
Konjugationsprobleme		
Konjugationsproblem (CP)	x, gxg^{-1}	g
Conjugate Decision Problem (CDP)	x, y	gibt es ein g mit $y = gxg^{-1}$?
Kombinierte Probleme		
Diskreter Logarithmus Konjugationsproblem (DLCP)	$g, x, g^a x g^{-a}$	a
Computational Diffie-Hellman Konjugationsproblem (CDHCP)	$g, x, g^a x g^{-a}, g^b x g^{-b}$	$g^{ab} x g^{-ab}$
Decisional Diffie-Hellman Konjugationsproblem (DDHCP)	$g, x, g^a x g^{-a}$, $g^b x g^{-b}, g^r x g^{-r}$	$r \stackrel{?}{=} ab$
CDH-type Konjugationsproblem	$x, g_1 x g_1^{-1}, g_2 x g_2^{-1}$	$g_1 g_2 x g_2^{-1} g_1^{-1}$
DDH-type Konjugationsproblem	$x, g_1 x g_1^{-1}, g_2 x g_2^{-1}, g_3 x g_3^{-1}$	$g_3 \stackrel{?}{=} g_1 g_2$
Exponentiation Problem (EP)	gxg^{-1}, a	$g^a x g^{-a}$

Tabelle 1: Kryptografische Probleme

7.1.1 Das DL, CDH und DDH Konjugationsproblem

In diesem Abschnitt befassen wir uns mit dem Diskreter Logarithmus Konjugationsproblem und den Diffie-Hellman Konjugationsproblemen. Wir untersuchen zunächst das DLCP. Einige Ergebnisse wie die eindeutige Lösbarkeit lassen sich direkt auf das CDHCP und das DDHCP übertragen.

Da das Konjugationsproblem keine eindeutige Lösung hat, muss auch das DLCP nicht unbedingt eine eindeutige Lösung haben. Seien etwa $g, x \in G$ so gewählt, dass $g^k \in Z(x)$ für ein $k \not\equiv 0 \pmod{\mathrm{ord}(g)}$ ist, und sei mit $g, x, g^a x g^{-a}$ eine Instanz des DLCP in G gegeben. Dann gilt $g^{a+n\cdot k} x g^{-(a+n\cdot k)} = g^a x g^{-a}$ für alle $n \in \mathbb{Z}$. In diesem Fall wären außer a auch $a + n \cdot k$ für $n \in \mathbb{Z}$ Lösung dieser Instanz.

Ein solcher Fall kann für die in Abschnitt 7.1 beschriebene Wahl von $g, x \in G$ nicht auftreten. Bevor wir diese Aussage beweisen, werden wir das DLCP, das CDHCP und das DDHCP noch einmal sauber definieren.

Definition 25 *Sei G eine nicht-abelsche Gruppe und $g, x \in G$ mit* $\text{ord}(g) = p$, *p prim, und $x \notin Z(g)$.*

- ***Diskreter Logarithmus Konjugationsproblem (DLCP):*** *Gegeben g, x und $g^a x g^{-a}$ für ein $a \in \mathbb{Z}$. Finde a.*
- ***Computational Diffie-Hellman Konjugationsproblem (CDHCP):*** *Gegeben g, x, $g^a x g^{-a}$ und $g^b x g^{-b}$. Finde $g^{ab} x g^{-ab}$.*
- ***Decisional Diffie-Hellman Konjugationsproblem (DDHCP):*** *Gegeben g, x, $g^a x g^{-a}$, $g^b x g^{-b}$ und $g^r x g^{-r}$. Entscheide, ob $r = ab$.*

Satz 28 *Sei G eine nicht-abelsche Gruppe und $g, x \in G$ mit* $\text{ord}(g) = p$, *p prim, und $x \notin Z(g)$. Dann hat die Instanz $g, x, g^a x g^{-a}$ des DLCP in G eine eindeutige Lösung modulo p.*

Beweis: Sei $b \in \mathbb{Z}$ eine Lösung der Instanz $g, x, g^a x g^{-a}$ des DLCP in G, d.h. es gilt $g^a x g^{-a} = g^b x g^{-b} \Leftrightarrow g^{a-b} x = x g^{a-b} \Leftrightarrow g^{ab} \in Z(x)$. In der Definition des DLCP ist gefordert, dass $\text{ord}(g) = p$, p prim, und $x \notin Z(g)$.
Angenommen $a \not\equiv b \pmod{p}$. Sei c das multiplikativ Inverse von $a - b \in \mathbb{Z}_p$. Dann folgt, dass $(g^{a-b})^c = g \in Z(x)$, und man erhält einen Widerspruch zu den Voraussetzungen. Also folgt $a \equiv b \pmod{p}$. □

Damit ist auch gezeigt, dass das CDHCP und das DDHCP eindeutige Lösungen haben, da sich Instanzen dieser Probleme in mehrere Instanzen des DLCP zerlegen lassen, die jeweils eine eindeutige Lösung haben.

Der folgende Satz sagt aus, dass das DLCP mindestens so schwer ist wie das DLP. Wir werden später sehen, dass es auch Gruppen gibt, in denen das DLP genauso schwer ist wie das DLCP. Analoge Aussagen lassen sich für die Diffie-Hellman Konjugationsprobleme zeigen.

Satz 29 *Sei G eine nicht-abelsche Gruppe und $\mathcal{A}$ ein Algorithmus, der das DLCP in G löst. Dann kann $\mathcal{A}$ dazu benutzt werden, um Instanzen g, g^a des DLP mit $g \notin Z(G)$ und* $\text{ord}(g)$ *prim in G zu lösen.*

Beweis: Sei g, g^a eine Instanz des DLP mit $g \notin Z(G)$ und $\text{ord}(g)$ prim in G. Der folgende Algorithmus $\mathcal{A}^*$ benutzt Algorithmus $\mathcal{A}$ als Unterprogramm und kann dazu benutzt werden, um Instanzen dieser Art des DLP in G zu lösen:

- Wähle ein $x \in G$ mit $x \notin Z(g)$.[17]
- Berechne $g^a x g^{-a}$.
- Gib das Tupel $(g, x, g^a x g^{-a})$ als Eingabe an $\mathcal{A}$.
- Mache die selbe Ausgabe wie $\mathcal{A}$.

Der Wert a ist eine Lösung der Instanz $(g, x, g^a x g^{-a})$ des DLCP in G. Satz 28 sagt aus, dass a die einzige ist. Algorithmus $\mathcal{A}^*$ macht deshalb genau dann die richtige Ausgabe, wenn $\mathcal{A}$ die richtige Ausgabe macht. □

Das DLCP hat die Eigenschaft, dass ein Angreifer, der das CP und das DLP in G lösen kann, damit noch nicht in der Lage ist, das DLCP in G zu lösen. Sei $g, x, g^a x g^{-a}$ eine Instanz des DLCP in G und L die Lösungsmenge der Instanz $x, g^a x g^{-a}$ des CP in G. Mit Satz 28 folgt, dass es genau ein $h \in L$ gibt, so dass g, h eine Instanz des DLP in G ist. Um den Wert a als Lösung des DLP zu berechnen, muss der Angreifer also zunächst den richtigen Wert $h \in L$ finden. Um ein sukzessives Ausprobieren zu verhindern, sollte die benutzte Gruppe G so gewählt werden, dass das Zentrum $Z(G)$ hinreichend groß ist. Idealerweise sollte $|Z(G)|$ exponentiell im Sicherheitsparameter sein.

Die aus Abschnitt 2.3 bekannte Hierarchie DDHP $\leq_P$ CDHP $\leq_P$ DLP spiegelt sich in ihren Kombinationen mit dem Konjugationsproblem wieder:

Satz 30 *Sei G eine nicht-abelsche Gruppe und $g, x \in G$ mit $\text{ord}(g) = p$, p prim, und $x \notin Z(g)$. Dann gilt*

$$DDHCP \leq_P CDHCP \leq_P DLCP.$$

Beweis: Die Aussage folgt aus Satz 28 über die eindeutige Lösbarkeit des DLCP. Sei mit $g, x, g^a x g^{-a}, g^b x g^{-b}$ eine Instanz des CDHCP in G gegeben. Durch Lösen der Instanzen $g, x, g^a x g^{-a}$ und $g, x, g^b x g^{-b}$ des DLCP in G erhält man die Werte a und b und kann $g^{ab} x g^{-ab}$ berechnen. Also gilt CDHCP $\leq_P$ DLCP.
Sei mit $g, x, g^a x g^{-a}, g^b x g^{-b}, g^r x g^{-r}$ eine Instanz des DDHCP gegeben. Lösen der Instanz $g, x, g^a x g^{-a}, g^b x g^{-b}$ des CDHCP liefert $g^{ab} x g^{-ab}$. Wegen der eindeutigen Lösbarkeit des DLCP gilt $r = ab$ genau dann, wenn $g^{ab} x g^{-ab} = g^r x g^{-r}$ erfüllt ist. Da letzteres effizient überprüft werden kann, gilt auch DDHCP $\leq_P$ CDHCP. □

[17] Damit die Reduktion polynomielle Laufzeit hat, muss es in G also insbesondere möglich sein, für ein gegebenes Gruppenelement $g \in G$ ein Element $x \in G$ mit $x \notin Z(g)$ in polynomieller Laufzeit zu finden.

Der folgende Satz sagt aus, dass in der Gruppe $GL(2, \mathbb{Z}_p)$ das DLP und CDH genauso schwer sind wie ihre Gegenstücke DLCP und CDHCP. Dabei nutzt man aus, dass man mit $g \in G$ ein Element aus dem Zentralisator des konjugierenden Elements g^a gegeben hat. Dieses kann benutzt werden, um aus der Instanz $g, x, g^a x g^{-a}$ des DLCP in $GL(2, \mathbb{Z}_p)$ Gruppenelemente der Form $g^a \cdot z$ mit $z \in Z(GL(2, \mathbb{Z}_p))$ zu berechnen.

Satz 31 *In $GL(2, \mathbb{Z}_p)$ gilt:*

- *DLCP $\Leftrightarrow_P$ DLP*
- *CDHCP $\Leftrightarrow_P$ CDHP*

Beweis: Wir zeigen zunächst, dass DLP $\leq_P$ DLCP gilt: Sei g, g^a eine Instanz des DLP in $GL(2, \mathbb{Z}_p)$. Wähle $x \in GL(2, \mathbb{Z}_p)$ mit $x \notin Z(g)$. Dann kann der gesuchte diskrete Logarithmus durch Lösen der Instanz $g, x, g^a x g^{-a}$ des DLCP in $SL(2, \mathbb{Z}_p)$ berechnet werden.
Nun zeigen wir, dass auch DLCP $\leq_P$ DLP gilt: Sei mit $g, x, g^a x g^{-a}$ eine Instanz des DLCP in $GL(2, \mathbb{Z}_p)$ gegeben. Mit $x, g^a x g^{-a}$ und $gx, g(g^a x g^{-a}) = g^a (gx) g^{-a}$ kann man daraus zwei Instanzen des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ extrahieren. Da $x \notin Z(gx)$ gilt, reichen diese Instanzen bereits aus, um das spezielle Konjugationsproblem zu lösen, d.h. ein Gruppenelement $\hat{g} \in GL(2, \mathbb{Z}_p)$ mit $Inn(\hat{g}) = Inn(g^a)$ zu berechnen (siehe Abschnitt 2.7.4).
Wir betrachten die Fälle, das $\det(g)$ ein quadratischer Rest bzw. ein quadratischer Nichtrest modulo p ist getrennt:
1. Fall: $\det(g) \in QR_p$. Sei $q_1 \in \mathbb{Z}_p$ eine Quadratwurzel von $\det(g)$ (diese kann effizient berechnet werden. Siehe etwa [35]). Dann ist $(q_1)^{-1} \cdot g \in SL(2, \mathbb{Z}_p)$. Es gilt ferner $Inn(g) = Inn((q_1)^{-1} \cdot g)$.
Aus $\det(g) \in QR_p$ folgt, dass auch $\det(g^a) = (\det(g))^a \in QR_p$ gilt. Da $Inn(g^a) = Inn(\hat{g})$ gilt, muss $\hat{g}$ von der Form $\hat{g} = g^a \cdot z$ mit $z \in Z(GL(2, \mathbb{Z}_p))$ sein. Das Zentrum von $GL(2, \mathbb{Z}_p)$ hat die Form $Z(GL(2, \mathbb{Z}_p)) = \{c \cdot I \mid c \in \mathbb{Z}_p^*\}$. Deshalb gilt $\det(z) \in QR_p$ für alle $z \in Z(GL(2, \mathbb{Z}_p))$. Damit ist auch $\det(\hat{g}) = \det(g^a \cdot z) = \det(g^a) \cdot \det(z)$ ein quadratischer Rest modulo p.
Sei $q_2 \in \mathbb{Z}_p$ eine Quadratwurzel von $\det(\hat{g})$. Wieder gilt $(q_2)^{-1} \cdot \hat{g} \in SL(2, \mathbb{Z}_p)$. Insgesamt ist $Inn(g^a) = Inn(q_2 \cdot \hat{g}) = Inn((q_1 \cdot g)^a)$ erfüllt. Da das Zentrum von $SL(2, \mathbb{Z}_p)$ nur $\pm I$ enthält, gilt $(q_1 \cdot g)^a = \pm q_2 \cdot \hat{g}$. Die Lösung der Instanz $g, x, g^a x g^{-a}$ des DLCP erhält man durch Lösen der Instanzen $q_1 \cdot g, \pm q_2 \cdot \hat{g}$ des DLP.
2. Fall: $\det(g) \notin QR_p$ geht man von g und $\hat{g}$ zunächst zu deren Quadraten $g^2 \bmod p$ und $\hat{g}^2 \bmod p$ über und verfährt ansonsten wie im Fall $\det(g) \in QR_p$.

Die Reduktion für das Computational Diffie-Hellman Problem verläuft analog. □

Das folgende Diagramm fasst die Situation in $GL(2, \mathbb{Z}_p)$ noch einmal zusammen. Dabei bedeutet $P_1 \Rightarrow P_2$, dass P_2 lösbar ist, wenn P_1 lösbar ist:

$$\begin{array}{ccc} \text{DLCP} & \Leftrightarrow & \text{DLP} \\ \Downarrow & & \Downarrow \\ \text{CDHCP} & \Leftrightarrow & \text{CDH} \\ \Downarrow & & \Downarrow \\ \text{DDHCP} & \Leftrightarrow & \text{DDH} \end{array}$$

7.1.2 Die DH-type Konjugationsprobleme

In der in Tabelle 1 definierten Form haben die DH-type Konjugationsprobleme keine eindeutige Lösung. Sei mit $x, g_1 x g_1^{-1}, g_2 x g_2^{-1}$ eine Instanz des CHD-type CP gegeben. Dann haben die Instanzen $x, g_1 x g_1^{-1}$ und $x, g_2 x g_2^{-1}$ des Konjugationsproblems mit $g_1 \cdot Z(x)$ bzw. $g_2 \cdot Z(x)$ jeweils $|Z(x)|$ Lösungen. Mit $g_1 z g_2 x g_2^{-1} z^{-1} g_1^{-1}$, wobei $z \in Z(x)$ kann das CDH-type CP also bis zu $\frac{|Z(x)|}{|Z(G)|}$ Lösungen haben.
Durch eine Instanz $x, g_1 x g_1^{-1}, g_2 x g_2^{-1}, g_3 x g_3^{-1}$ des DDH-type CP sind die Werte g_1, g_2 und g_3 ebenfalls nicht eindeutig bestimmt. Das DDH-type CP besteht also eigentlich darin, zu entscheiden, ob Elemente $\hat{g}_1, \hat{g}_2, \hat{g}_3 \in G$ mit $g_i x g_i^{-1} = \hat{g}_i x \hat{g}_i^{-1}$ für $1 \leq i \leq 3$ und $\hat{g}_1 \cdot \hat{g}_2 = \hat{g}_3$ existieren. Formal definieren wir das CDH-type CP und das DDH-type CP wie folgt:

Definition 26 *Sei G eine nicht-abelsche Gruppe und $x \in G$. Wir definieren die Abbildung $i_x : G \to Inn(G)(x)$, $g \mapsto gxg^{-1}$, wobei $Inn(G)(x) = \{gxg^{-1} \mid g \in G\}$. Das Urbild $i_x^{-1}(g)$ eines Elements $g \in Inn(G)(x)$ unter i_x entspricht der Lösungsmenge der Instanz x, g des Konjugationsproblems in G. Wir definieren:*

- ***CDH-type Konjugationsproblem (CDH-type CP):*** *Gegeben seien $x \in G$ und $g_1, g_2 \in Inn(G)(x)$. Finde ein $g_3 \in Inn(G)(x)$ mit $i_x^{-1}(g_3) \cap i_x^{-1}(g_1) \cdot i_x^{-1}(g_2) \neq \emptyset$.*
- ***CDH-type Konjugationsproblem (CDH-type CP):*** *Gegeben seien $x \in G$ und $g_1, g_2, g_3 \in Inn(G)(x)$. Entscheide, ob $i_x^{-1}(g_3) \cap i_x^{-1}(g_1) \cdot i_x^{-1}(g_2) = \emptyset$.*

Der einfacheren Handhabbarkeit wegen werden wir Instanzen des CDH-type und des DDH-type Konjugationsproblems weiter in der Form angeben, die in Tabelle 1 benutzt wurde, d.h. durch Angabe der Gruppenelemente $x, g_1 x g_1^{-1}, g_2 x g_2^{-1}$ bzw. $x, g_1 x g_1^{-1}, g_2 x g_2^{-1}$, $g_3 x g_3^{-1}$. Der folgende Satz verdeutlicht die enge Verwandtschaft zwischen dem CDH-type CP und dem CP.

Satz 32 *Sei G eine nicht-abelsche Gruppe. Dann gilt*

$$\textit{CDH-type CP} \leq_P \textit{CP}.$$

Beweis: Sei mit $x, g_1 x g_1^{-1}, g_2 x g_2^{-1}$ eine Instanz des CDH-type CP in G gegeben. Ist $\hat{g}_1$ eine Lösung der Instanz $x, g_1 x g_1^{-1}$ des CP in G, dann ist $\hat{g}_1 g_2 x g_2^{-1} \hat{g}_1^{-1}$ eine Lösung des CDH-type CP. □

Das DDH-type CP lässt sich nicht so einfach auf das CP reduzieren: Sei $x, g_1xg_1^{-1}, g_2xg_2^{-1}$, $g_3xg_3^{-1}$ eine Instanz des DDH-type CP mit $g_3 = g_1 \cdot g_2$. Seien ferner $\hat{g}_1, \hat{g}_2, \hat{g}_3 \in G$ Lösungen der Instanzen $x, g_1xg_1^{-1}$, $x, g_2xg_2^{-1}$ bzw. $x, g_3xg_3^{-1}$ des Konjugationsproblems in G. Dann muss nicht notwendigerweise $\hat{g}_1 \cdot \hat{g}_2 = \hat{g}_3$ gelten.

Die nicht-eindeutige Lösbarkeit des CDH-type CP kann man sich für die Konstruktion kryptografischer Verfahren zunutze machen. Der Ansatz ist dabei ähnlich wie beim DLP in $Inn(G)$ (siehe Abschnitt 3.1). Man wählt eine nicht-abelsche Gruppe G mit einem hinreichend großen Zentrum. Dann haben das Konjugationsproblem und das CDH-type CP eine große Lösungsmenge. Ein Angreifer, der das Konjugationsproblem in G lösen kann, kann auch Lösungen des CDH-type CP berechnen. Die verwendeten kryptografischen Verfahren werden nun so konstruiert, dass ein Angreifer eine bestimmte Lösung kennen muss, um erfolgreich zu sein. Das kann zum Beispiel dadurch erreicht werden, dass ein Teilnehmer das konjugierende Element g_1 als geheimen Schlüssel wählt und die Elemente x und $g_1xg_1^{-1}$ als öffentlichen Schlüssel preisgibt. Während der Ausführung des kryptografischen Verfahrens wird dann ein weiterer Wert $g_2xg_2^{-1}$ erzeugt. Die Werte x, $g_1xg_1^{-1}$ und $g_2xg_2^{-1}$ bilden zusammen eine Instanz des CDH-type CP in G. Um sein Ziel zu erreichen (etwa einen Geheimtext zu entschlüsseln), muss dem Angreifer der Wert $g_1g_2xg_2^{-1}g_1^{-1}$ bekannt sein. Der Besitzer des geheimen Schlüssels g_1 kann den gesuchten Wert effizient berechnen. Für einen Angreifer, der das CP lösen kann und g_1 nicht kennt, sehen alle $|Z(x)|$ Lösungen der Instanz $x, g_1xg_1^{-1}$ jedoch gleich aus und es bleibt ihm nur die vollständige Suche, um sein Ziel zu erreichen. Die Gruppe G sowie die Elemente x, g_1, g_2 müssen in diesem Fall so gewählt werden, dass eine vollständige Suche praktisch nicht durchführbar ist. Dieser Ansatz wird in den Umsetzungen des Diffie-Hellman Protokolls und der ElGamal-Verschlüsselung auf nicht-abelschen Gruppen aus [23] verwendet (siehe auch Abschnitt 7.2).

In den in [23] vorgestellten Protokollen werden die konjugierenden Elemente aus kommutierenden Untergruppen $A, B \subseteq G$ gewählt, d.h. für $a \in A$ und $b \in B$ gilt $ab = ba$. Das CDH-type CP ist in diesen Fällen von der Form x, axa^{-1}, bxb^{-1} mit $a \in A$ und $b \in B$. Durch die spezielle Wahl der konjugierenden Elemente a, b ist sichergestellt, dass das CDH-type CP eine eindeutige Lösung hat: Seien $\hat{a} \in A$, $\hat{b} \in B$ mit $\hat{a}x\hat{a}^{-1} = axa^{-1}$ und $\hat{b}x\hat{b}^{-1} = bxb^{-1}$. Dann gibt es $z_1, z_2 \in Z(x)$ mit $\hat{a} = a \cdot z_1$ und $\hat{b} = b \cdot z_2$ und es gilt:

$$\begin{aligned}
\hat{a}\hat{b}x\hat{b}^{-1}\hat{a}^{-1} &= \hat{a}bz_2xz_2^{-1}b^{-1}\hat{a}^{-1} \\
&= \hat{a}bxb^{-1}\hat{a}^{-1} \\
&= b\hat{a}x\hat{a}^{-1}b^{-1} \\
&= baz_1xz_1^{-1}a^{-1}b^{-1} \\
&= baxa^{-1}b^{-1}.
\end{aligned}$$

Da das Konjugationsproblem in $GL(2, \mathbb{Z}_p)$ effizient lösbar ist, können auch Lösungen des CDH-type CP in $GL(2, \mathbb{Z}_p)$ effizient berechnet werden. Wir zeigen nun, dass ein Angreifer auch die Version des CDH-type CP, bei der die konjugierenden Elemente aus kommutierenden Untergruppen A und B gewählt wurden, in $GL(2, \mathbb{Z}_p)$ effizient lösen kann:

Seien A, B kommutierende Untergruppen und x, axa^{-1}, bxb^{-1} mit $a \in A$ und $b \in B$ eine Instanz des CDH-type CP in $GL(2, \mathbb{Z}_p)$. Das Paar x, axa^{-1} bildet eine Instanz des CP in $GL(2, \mathbb{Z}_p)$. Neben dieser Instanz kennt ein Angreifer auch Elemente aus $Z(a)$ (die Elemente aus B erfüllen diese Eigenschaft). Für ein beliebiges $b \in B$ erhält er mit x, axa^{-1} und $x, b \cdot axa^{-1} = a(bx)a^{-1}$ zwei simultane Instanzen des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ mit $x \notin Z(bx)$. Wie wir bereits in Abschnitt 2.7.4 gesehen haben, reicht das aus, um das spezielle Konjugationsproblem in $GL(2, \mathbb{Z}_p)$ zu lösen und einen Wert $\hat{a} \in G$ mit $Inn(a) = Inn(\hat{a})$ zu berechnen. Der Wert $\hat{a}$ ist von der Form $\hat{a} = a \cdot z$ mit $z \in Z(GL(2, \mathbb{Z}_p))$ und damit zum geheimen Schlüssel a gleichwertig. Durch Konjugation von bxb^{-1} mit $\hat{a}$ erhält man mit $\hat{a}bxb^{-1}\hat{a}^{-1} = azbxb^{-1}z^{-1}a^{-1} = abxb^{-1}a^{-1}$ die gesuchte Lösung des CDH-type CP.

Ähnlich sieht es mit dem DDH-type CP aus. Sind außer $x, g_1xg_1^{-1}, g_2xg_2^{-1}, g_3xg_3^{-1}$ Elemente aus den Zentralisatoren von g_1, g_2 bzw. g_3 bekannt, so können $\hat{g}_1, \hat{g}_2, \hat{g}_3 \in G$ berechnet werden mit $Inn(g_i) = Inn(\hat{g}_i)$ für $1 \leq i \leq 3$. Nun kann man leicht testen, ob $\hat{g}_1 \cdot \hat{g}_2$ und $\hat{g}_3$ sich lediglich um ein Element aus $Z(G)$ unterscheiden.
Sind keine Elemente aus den Zentralisatoren von g_1, g_2 bzw. g_3 bekannt, so nutzt man aus, dass man in der Gruppe $GL(2, \mathbb{Z}_p)$ nicht nur einzelne Lösungen des Konjugationsproblems sondern die komplette Lösungsmenge berechnen kann (siehe Abschnitt 2.7.4). Wir geben nun ein konstruktives Verfahren an, dass das DDH-type CP in $GL(2, \mathbb{Z}_p)$ löst. Dabei versuchen wir, Gruppenelemente $\hat{g}_1, \hat{g}_2, \hat{g}_3 \in G$ zu berechnen mit $\hat{g}_1 \cdot \hat{g}_2 = \hat{g}_3$ und $g_ixg_i^{-1} = \hat{g}_i x \hat{g}_i^{-1}$ für $1 \leq i \leq 3$, d.h. wir suchen Elemente $z_i \in Z(x)$ mit $\hat{g}_i = g_i \cdot z_i$ für $1 \leq i \leq 3$ und $g_1z_1 \cdot g_2z_2 = g_3z_3$.
Sei $\bar{g}_3$ eine beliebige Lösung der Instanz $x, g_3xg_3^{-1}$ des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$. Wir wissen, dass es ein $z_4 \in Z(x)$ gibt, mit $g_3z_4 = \bar{g}_3$. Die zu betrachtende Gleichung vereinfacht sich deshalb zu $g_1z_1 \cdot g_2z_5 = \hat{g}_1 \cdot \hat{g}_2 = \bar{g}_3$ mit $z_5 = z_3z_4^{-1} \in Z(x)$. Für $\hat{g}_1$ kann man o.B.d.A. annehmen, dass $\det(\hat{g}_1) \in \{1, n\}$, wobei $n \in \mathbb{Z}_p$ ein beliebiger quadratischer Nichtrest modulo p ist: Falls $\det(\hat{g}_1)$ ein quadratischer Rest (bzw. Nichtrest) modulo p ist, existiert nämlich ein $c \in \mathbb{Z}_p$ mit $\det(c\hat{g}_1) = 1$ (bzw. $\det(c\hat{g}_1) = n$). Da $c \cdot I \in Z(GL(2, \mathbb{Z}_p))$ ist, fällt dieser Faktor bei der Konjugation weg. Für $\hat{g}_1$ und $\hat{g}_2$ setzt man nun die in Abschnitt 2.7.4 hergeleiteten allgemeinen Lösungen der Instanzen $x, g_1xg_1^{-1}$ und $x, g_2xg_2^{-1}$ des Konjugationsproblems ein (diese hängen jeweils von 2 freien Variablen ab). Mit unserer Annahme über die Determinante von $\hat{g}_1 = g_1z_1$ kann man das Inverse von $\hat{g}_1$ berechnen und die Gleichung $\hat{g}_1\hat{g}_2 = \bar{g}_3$ umformen zu $\hat{g}_2 = \bar{g}_3\hat{g}_1^{-1}$. Durch Koeffizientenvergleich erhält man ein lineares Gleichungssystem, durch dessen Lösung man die gesuchten $\hat{g}_1, \hat{g}_2 \in GL(2, \mathbb{Z}_p)$ berechnen kann. Damit wurde der folgende Satz bewiesen.

Satz 33 *In $GL(2, \mathbb{Z}_p)$ sind das CDH-type CP und das DDH-type CP effizient lösbar. Das gilt insbesondere auch, wenn die konjugierenden Elemente aus kommutierenden Untergruppen gewählt werden.*

7.1.3 Eine DLP-ähnliche Einwegfunktion

Einwegfunktionen spielen beim Design asymmetrischer kryptografischer Verfahren und von kryptografischen Protokollen eine zentrale Rolle. Dieser Abschnitt soll zunächst zeigen, dass auf nicht-abelschen Gruppen G interessante Einwegfunktionen definiert werden können. Wir verwenden zur Konstruktion der Einwegfunktion wieder eine Kombination aus DLP und CP. Eine Variante dieser Einwegfunktionen wird in Abschnitt 7.3 als Grundlage zur Konstruktion eines sicheren Commitment Schemes verwendet.

Definition 27 *Sei G eine nicht-abelsche Gruppe und seien $g, h \in G$ mit $g \notin Z(h)$ und $ord(g) = p$, $ord(h) = q$ mit p, q prim. Die Abbildung $F_{g,h}$ wird definiert durch:*

$$\begin{aligned} F_{g,h} : \mathbb{Z}_p \times \mathbb{Z}_q &\rightarrow G \\ (x, y) &\mapsto g^x h^y g^{-x}. \end{aligned}$$

Bemerkungen:

- Wendet man diese Konstruktion auf eine abelsche Gruppe an oder wählt man $g \in Z(h)$, so stellt $F_{g,h}$ die diskrete Exponentiation dar:

$$\begin{aligned} F_{g,h} : \mathbb{Z}_p \times \mathbb{Z}_q &\rightarrow G \\ (x, y) &\mapsto h^y \end{aligned}$$

- Für feste x erhält man eine homomorphe Funktion, d.h. für $x \in \mathbb{Z}_p$, $y, z \in \mathbb{Z}_q$ und $k \in \mathbb{Z}$ gilt

$$\begin{aligned} F_{g,h}(x, y) \cdot F_{g,h}(x, z) &= F_{g,h}(x, y + z) \quad \text{und} \\ F_{g,h}(x, y)^k &= F_{g,h}(x, k \cdot y). \end{aligned}$$

- Für beliebige $r_1, r_2 \in \mathbb{Z}_p$ gilt $F_{g,h}(r_1, 0) = F_{g,h}(r_2, 0)$. Die Funktion $F_{g,h}$ ist also in der definierten Form nicht injektiv. Das folgende Lemma sagt aus, dass man die Injektivität von $F_{g,h}$ erhält, wenn man fordert, dass das zweite Argument der Funktion ungleich Null ist.

  Lemma 5 *Seien $g, h \in G$ mit $g \notin Z(h)$, $ghg^{-1} \notin Z(h)$ und $ord(g) = p$, $ord(h) = q$ mit p, q prim. Schränkt man den Definitionsbereich von $F_{g,h}$ auf $\mathbb{Z}_p \times \mathbb{Z}_q^*$ ein, so erhält man mit*

$$\begin{aligned} F_{g,h} : \mathbb{Z}_p \times \mathbb{Z}_q^* &\rightarrow G \\ (x, y) &\mapsto g^x h^y g^{-x} \end{aligned}$$

  eine injektive Funktion.

Beweis: Seien $r_1, \hat{r}_1 \in \mathbb{Z}_p$ und $r_2, \hat{r}_2 \in \mathbb{Z}_q^*$ mit $F_{g,h}(r_1, r_2) = F_{g,h}(\hat{r}_1, \hat{r}_2)$, d.h. $g^{r_1} h^{r_2} g^{-r_1} = g^{\hat{r}_1} h^{\hat{r}_2} g^{-\hat{r}_1}$. Sei $\bar{r} = r_1 - \hat{r}_1$. Dann gilt $g^{\bar{r}} h^{r_2} g^{-\bar{r}} = h^{\hat{r}_2}$. Für alle $k \in \mathbb{Z}$ folgt, dass $g^{\bar{r}} h^{k \cdot r_2} g^{-\bar{r}} = h^{k \cdot \hat{r}_2}$ gilt. Wegen der Wahl von $r_2, \hat{r}_2 \in \mathbb{Z}_q^*$ folgt, dass $Inn(g^{\bar{r}})$ eine Permutation auf $< h >$ ist.
Wir nehmen nun an, dass $r_1 \not\equiv \hat{r}_1 \pmod{p}$ und damit $\bar{r} \not\equiv 0 \pmod{p}$ gilt. Sei $\hat{r}$ das multiplikativ Inverse von $\bar{r}$ modulo p. Dann ist die Abbildung $(Inn(g^{\bar{r}}))^{\hat{r}} = Inn((g^{\bar{r}})^{\hat{r}}) = Inn(g)$ ebenfalls eine Permutation auf $< h >$. Insbesondere folgt, dass $Inn(g)(h) = ghg^{-1} \in < h > \subseteq Z(h)$ gilt, was ein Widerspruch zu den Voraussetzungen des Satzes ist. Der Fall $r_1 \not\equiv \hat{r}_1 \pmod{p}$ kann also nicht eintreten. Aus $r_1 \equiv \hat{r}_1 \pmod{p}$ folgt, dass dann auch $h^{r_2} = h^{\hat{r}_2}$ und damit $r_2 \equiv \hat{r}_2 \pmod{q}$ gilt. □

- Die Funktion $F_{g,h}$ verfügt über für die Konstruktion kryptografischer Protokolle nützliche Blendungseigenschaften. Seien $x \in \mathbb{Z}_p$ und $y \in \mathbb{Z}_q^*$. Für zufällig gewählte $r_1 \in \mathbb{Z}_p$ und $r_2 \in \mathbb{Z}_q^*$ sind die Werte $x + r_1$ und $y \cdot r_2$ uniform verteilt in $\mathbb{Z}_p$ bzw. $\mathbb{Z}_q^*$. Mit Lemma 5 folgt, dass der Wert

$$g^{r_1}(F_{g,h}(x, y))^{r_2} g^{-r_1} = F_{g,h}(x + r_1, y \cdot r_2)$$

  uniform in $F_{g,h}(\mathbb{Z}_p, \mathbb{Z}_q^*)$ verteilt ist.

  Bezeichne

$$\begin{aligned} B : F_{g,h}(\mathbb{Z}_p, \mathbb{Z}_q) \times \mathbb{Z}_p \times \mathbb{Z}_q &\rightarrow F_{g,h}(\mathbb{Z}_p, \mathbb{Z}_q) \\ (x, r_1, r_2) &\mapsto g^{r_1} x^{r_2} g^{-r_1} \end{aligned}$$

  die Blendfunktion für $F_{g,h}$.Dann gilt:

  - $B(B(x, r_1, r_2), -r_1, r_2^{-1} \pmod{q}) = x$ für $r_2 \neq 0 \pmod{q}$, d.h. eine Blendung mit (r_1, r_2) kann durch eine Blendung mit $(-r_1, r_2^{-1} \pmod{q})$ wieder rückgängig gemacht werden.
  - $B(B(x, r_1, r_2), r_3, r_4) = B(B(x, r_3, r_4), r_1, r_2)$, d.h. bei mehreren Blendungen ist die Reihenfolge, in der diese vorgenommen werden, beliebig.

Die Homomorphie- und Blendungseigenschaften werden oft für die Konstruktion kryptografischer Protokolle verwendet (siehe etwa [53]). Dabei stellt die Homomorphie-Eigenschaft sicher, dass auf Bildern der Funktion durchgeführte Berechnungen sich auf deren Urbilder übertragen. Teilnehmer des Protokolls können also Berechnungen auf Werten durchführen, die sie selbst nicht kennen.

Die folgenden Sätze stellen einen Zusammenhang zwischen dem Diskreter Logarithmus Problem in $< g >$ bzw. $< h >$ und der Invertierbarkeit der Funktion $F_{g,h}$ her.

Satz 34 *Wenn das DLP schwer ist in $< g >$ (bzw. $< h >$), so ist r_1 (bzw. r_2) ein Hard-Core Prädikat der Funktion $F_{g,h}$.*

Beweis: Sei $\mathcal{A}$ ein Angreifer, der für gegebene Funktionswerte $F_{g,h}(r_1, r_2)$ den Wert r_1 berechnen kann. Wir zeigen, dass dieser Angreifer benutzt werden kann, um diskrete Logarithmen in der Gruppe $< g >$ zu berechnen: Sei x, g^x eine Instanz des DLP in $< g >$. Wähle $r_2 \in_R \mathbb{Z}_q^*$ zufällig, gib $g^x h^{r_2} g^{-x}$ an $\mathcal{A}$ und mache die selbe Ausgabe wie $\mathcal{A}$.
Wegen der Injektivität von $F_{g,h}$ ist die Ausgabe dieses Algorithmus genau dann korrekt, wenn $\mathcal{A}$ den richtigen Wert ausgibt. Der Beweis, dass r_2 ein Hard-Core Prädikat von $F_{g,h}$ ist, wenn das DLP in $< h >$ schwer ist, verläuft vollkommen analog. □

Der letzte Satz sagt aus, dass es für ein gegebenes Bild der Funktion $F_{g,h}$ schwer ist, bestimmte Teile des Urbildes zu berechnen, sofern das DLP schwer ist in den Gruppen $< g >$ bzw. $< h >$. Daraus folgt direkt, dass unter der selben Annahme auch die Berechnung kompletter Urbilder der Funktion $F_{g,h}$ schwer ist.

Korollar 5 *Wenn das DLP schwer ist in $< g >$ oder $< h >$, so ist $F_{g,h}$ eine Einwegfunktion.*

Die folgenden beiden Sätze stellen einen Zusammenhang zwischen der Invertierbarkeit von $F_{g,h}$ und der Schwierigkeit des Konjugationsproblems her.

Satz 35 *Wenn Instanzen der Form $(h^{r_2}, F_{g,h}(r_1, r_2))$ des CP in G schwer sind, so ist g^{r_1} ein Hard-Core Prädikat der Funktion $F_{g,h}$.*

Beweis: Sei $h^{r_2}, F_{g,h}(r_1, r_2)$ eine Instanz der vorgegebenen Form des CP in G. Angenommen, einem Angreifer gelingt es, aus $F_{g,h}(r_1, r_2))$ den Wert g^{r_1} zu berechnen, so hat er vorliegende Instanz des CP gelöst. □

Da g öffentlich ist und damit g^{r_1} leicht aus r_1 berechnet werden kann, gilt auch das folgende Korollar.

Korollar 6 *Wenn Instanzen der Form $h^{r_2}, F_{g,h}(r_1, r_2)$ des CP in G schwer sind, so ist r_1 ein Hard-Core Prädikat der Funktion $F_{g,h}$.*

7.2 Kryptografische Protokolle auf nicht-abelschen Gruppen

In diesem Abschnitt werden wir mit zwei dem Diffie-Hellman Protokoll bzw. der ElGamal-Verschlüsselung ähnlichen Verfahren und dem Cayley-Purser Algorithmus kryptografische Verfahren auf nicht-abelschen Gruppen vorstellen, deren Sicherheit auf den im letzten Abschnitt definierten Problemen basiert.
Alle vorgestellten Protokolle lassen sich auch auf der Gruppe $GL(2, \mathbb{Z}_p)$ ausführen. Wir zeigen, dass diese Protokolle auf $GL(2, \mathbb{Z}_p)$ auf die selbe Weise angreifbar sind wie MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$. Wir arbeiten die gemeinsame Designschwäche dieser Protokolle heraus und zeigen, wie man sie umgehen kann. Die dabei erarbeiteten Designkriterien für kryptografische Protokolle auf $GL(2, \mathbb{Z}_p)$ werden wir im folgenden Abschnitt nutzen, um ein beweisbar sicheres Commitment Scheme auf $GL(2, \mathbb{Z}_p)$ zu entwerfen.

7.2.1 Verfahren basierend auf dem CDH-type Konjugationsproblem

Die in diesem Abschnitt behandelten Verfahren verwenden die bereits in Abschnitt 7.1.2 eingeführte Variante des CDH-type Konjugationsproblems auf kommutierenden Untergruppen als zugrundeliegende Sicherheitsannahme. Für die benutzte Gruppe G setzen wir also voraus, dass zwei hinreichend große nicht-triviale kommutierende Untergruppen $A, B \subseteq G$ existieren, d.h. für $a \in A$ und $b \in B$ gelte stets $ab = ba$. In G muss ferner das CDH-type Konjugationsproblem (siehe Tabelle 1 auf Seite 89) für diese speziellen Instanzen schwer sein, d.h. seien $x \in G$, $a \in A$ und $b \in B$ und die Elemente x, axa^{-1} und bxb^{-1} gegeben, so kann das Gruppenelement $abx(ab)^{-1} = bax(ba)^{-1}$ nicht effizient berechnet werden.

In [23] wurden ein Protokoll zur Schlüsselvereinbarung und ein asymmetrisches Verschlüsselungsverfahren vorgestellt, deren Sicherheit auf der Schwierigkeit dieser speziellen Instanzen des CDH-type Konjugationsproblems beruht. Das Protokoll zur Schlüsselvereinbarung erinnert stark an das Diffie-Hellman Protokoll [13] (siehe auch Abschnitt 2.4). Wie in abelschen Gruppen auch kann man das Schlüsselvereinbarungsprotokoll nutzen, um ein asymmetrisches Verschlüsselungsverfahren zu definieren, das dem ElGamal Schema sehr ähnlich ist.

Protokoll zur Schlüsselvereinbarung: Seien G eine nicht-abelsche Gruppe, $A, B \subseteq G$ kommutierende Untergruppen von G und $x \in G$. Das Protokoll besteht aus den folgenden zwei Schritten, die parallel ausgeführt werden können:

- Teilnehmer A wählt ein zufälliges $a \in_R A$ und sendet axa^{-1} an Teilnehmer B.
- Teilnehmer B wählt ein zufälliges $b \in_R B$ und sendet bxb^{-1} an Teilnehmer A.

Nach Erhalt der Nachricht kann jeder Teilnehmer das gemeinsame Geheimnis $k \in G$ durch Konjugation des erhaltenen Wertes mit dem selbst gewählten Gruppenelement berechnen: $k = b(axa^{-1})b^{-1} = a(bxb^{-1})a^{-1}$.

Ein passiver Angreifer kann genau dann den gemeinsamen Schlüssel k aus den abgehörten Werten axa^{-1} und bxb^{-1} berechnen, wenn er das CDH-type Konjugationsproblem für diese speziellen Instanzen lösen kann. Wie das Diffie-Hellman Protokoll ist auch diese Variante für nicht-abelsche Gruppen nicht sicher gegen aktive Angreifer. Beim Versand der Nachrichten sollte also zusätzlich die Authentizität der ausgetauschten Nachrichten abgesichert werden.

Asymmetrisches Verschlüsselungsverfahren: In zyklischen Gruppen kann man das Diffie-Hellman Protokoll nutzen, um ein semantisch sicheres asymmetrisches Verschlüsselungsverfahren zu konstruieren, das ElGamal-Verfahren (siehe Abschnitt 2.5). Dabei wird mittels des Diffie-Hellman Protokolls ein Sitzungsschlüssel vereinbart, der anschließend zur Blendung der übertragenen Nachricht benutzt wird. Verwendet man einen Hashwert des Sitzungsschlüssels als Blendfaktor, so können als Klartexte sogar beliebige Nachrichten (fester Länge) gewählt werden (siehe Abschnitt 2.5). Eine analoge Konstruktion kann auch für das vorgeschlagene Protokoll zur Schlüsselvereinbarung auf nicht-abelschen Gruppen vorgenommen werden. In diesem Fall wählt man eine nicht-abelsche Gruppe G und eine Hash-Funktion $H : G \to \{0,1\}^k$ als öffentliche Parameter. Jeder Teilnehmer wählt ferner ein $x \in G$ und ein zufälliges $a \in_R A$. Dabei bilden der Wert a den geheimen und die Werte x und axa^{-1} den öffentlichen Schlüssel dieses Teilnehmers.

Um eine Nachricht $m \in \{0,1\}^k$ zu verschlüsseln, wählt sich der Absender einen zufälligen Wert $b \in_R B$ und sendet das Paar (c,d), wobei $c = bxb^{-1}$ und $d = H(byb^{-1}) \oplus m$ ist, an den Empfänger. Aufgrund der Kommutativität der Untergruppen A und B gilt $aca^{-1} = abxb^{-1}a^{-1} = baxa^{-1}b^{-1} = byb^{-1}$. Um den erhaltenen Geheimtext zu entschlüsseln, berechnet der Empfänger $m = d \oplus H(aca^{-1})$.
Das Verfahren ist semantisch sicher im Random Oracle Modell unter der Annahme, dass das CDH-type Konjugationsproblem für die gewählten Instanzen schwer ist.

Umsetzung auf Zopfgruppen: Diese Protokolle wurden zunächst für Zopfgruppen vorgeschlagen [23]. Die n-Zopfgruppe B_n ($n \in \mathbb{N}$) ist durch folgende Präsentation definiert

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{für } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{für } |i-j| \geq 2 \end{array} \right\rangle$$

Die Zahl n heißt Zopfindex und jedes Element aus B_n ein n-Zopf. Die Gruppe B_n ist unendlich und nicht-abelsch. Anschaulich kann man sich die Elemente aus B_n anhand von n verflochtenen Strängen vorstellen, die zwischen zwei horizontalen Balken gespannt sind. Zwei Zöpfe sind äquivalent, wenn sie stetig ineinander überführt werden können (siehe Abbildung 1 auf der nächsten Seite). Das Produkt ab zweier Zöpfe a und b erhält man, wenn man Zopf a über Zopf b ansetzt. Dem neutralen Element von B_n entspricht in dieser geometrischen Interpretation der Zopf mit n vertikalen Strängen (ohne Verflechtung). Das Inverse eines Zopfes erhält man, wenn man ihn an einer horizontalen Geraden spiegelt.

In der geometrischen Interpretation beschreibt jeder n-Zopf eine Permutation π aus Σ_n, indem man den Endpunkt von Strang i als das Bild $\pi(i)$ von i unter der Permutation π

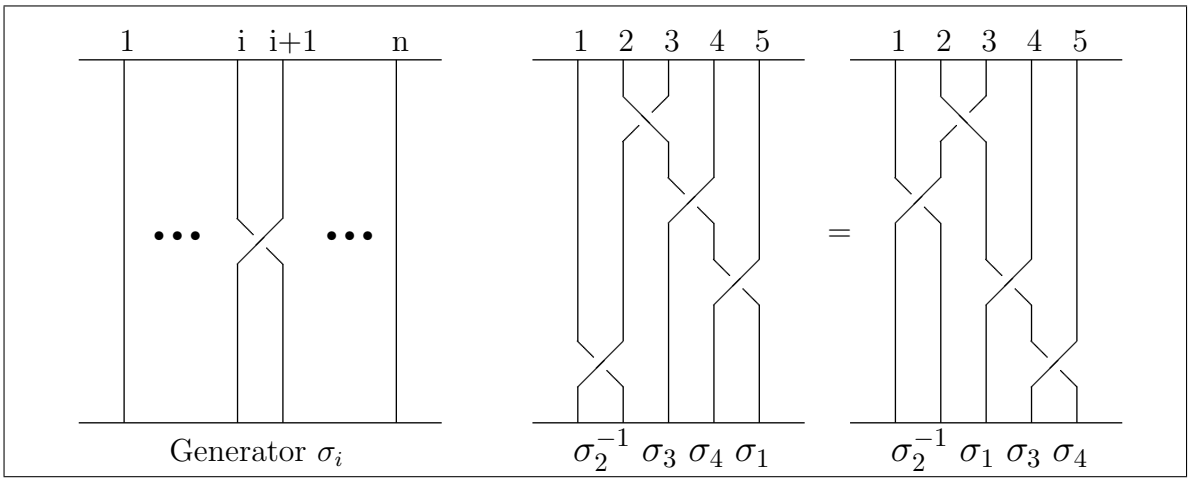

Abbildung 1: Geometrische Interpretation der Zopfgruppen

auffasst. Auf diese Weise erhält man einen Homomorphismus von B_n nach Σ_n. Fügt man in der Präsentation von B_n noch die Relation $\sigma_i^2 = 1$ für $i = 1, \ldots, n-1$ hinzu, so erhält man die symmetrische Gruppe Σ_n.

Zopfgruppen sind wegen ihrer Eigenschaften interessant für kryptografische Anwendungen:

- Mit dem Konjugationsproblem gibt es auf Zopfgruppen ein gut untersuchtes Problem, für das bis heute kein effizienter Algorithmus gefunden wurde.
- Für Elemente aus B_n gibt es eine kanonische Form, die sich effizient auf Computern umsetzen lässt. Diese Darstellungsweise erlaubt ferner eine effiziente Berechnung von Gruppenoperationen (siehe [8]).

Als Sicherheitsparameter dient die Anzahl der Stränge n.

Für die Konstruktion der vorgestellten kryptografischen Protokolle werden zwei kommutierende Untergruppen $A, B \subseteq G$ der benutzten nicht-abelschen Gruppe G benötigt. Für $G = B_n$ wählt man dafür LB_l und RB_r, wobei $n = l+r$. Die Untergruppe LB_l besteht dabei aus den n-Zöpfen, die nur die linken l Stränge verwenden, d.h. LB_l ist die von $\sigma_1, \ldots, \sigma_{l-1}$ erzeugte Untergruppe. Analog benutzt RB_r nur die rechten r Stränge, d.h. RB_r wird von $\sigma_{l+1}, \ldots, \sigma_{l+r-1}$ erzeugt. Die zweite Relation $\sigma_i\sigma_j = \sigma_j\sigma_i$für $|i-j| \geq 2$ aus der Präsentation von B_n garantiert die Kommutativität der beiden benutzten Untergruppen.

Einen Überblick über Kryptografie auf Zopfgruppen geben [7, 12]. Als Referenz für die Implementierung von Algorithmen auf Zopfgruppen kann [8] herangezogen werden. Kürzlich wurde von Cheon und Jun in [10] ein Algorithmus mit erwartet polynomieller Laufzeit

zur Lösung des CDH-type CP auf Zopfgruppen vorgestellt.[18] Die Laufzeit des Algorithmus ist mit $\mathcal{O}(n^{14.4}l^{3.2})$, wobei n der Index der benutzten Zopfgruppe und l die Charney Länge der benutzten Zöpfe ist, zwar noch zu hoch, um damit Instanzen des CDH-type CP angreifen zu können, wie sie in den aktuellen kryptografischen Verfahren auf Zopfgruppen auftreten. Die Tatsache, dass ein Algorithmus mit erwartet polynomieller Laufzeit existiert, der eventuell noch optimiert werden kann, schwächt jedoch das Vertrauen in diese Systeme.

Umsetzung auf Matrixgruppen: Die im letzten Abschnitt vorgestellten Protokolle lassen sich auch auf der Gruppe $GL(2, \mathbb{Z}_p)$ ausführen. In Abschnitt 7.1.2 wurde bereits gezeigt, dass das CDH-type CP in $GL(2, \mathbb{Z}_p)$ effizient lösbar ist (siehe Satz 33 auf Seite 95). Die Lösbarkeit des CDH-type CP kann wie folgt ausgenutzt werden, um die vorgestellten Protokolle auf $GL(2, \mathbb{Z}_p)$ anzugreifen:

Seien G eine nicht-abelsche Gruppe und $A, B \subseteq G$ kommutierende Untergruppen von G. Sei ferner x, axa^{-1} mit $a \in A$ eine Instanz des Konjugationsproblems, wie sie in den vorgestellten Protokollen auftritt. Wir nutzen nun aus, dass die Mengen A und B bekannt sind (mindestens eine von ihnen muss Teil des öffentlichen Schlüssels sein). Sei $\bar{b} \in B$ beliebig. Dann erhält man mit $\bar{b} \cdot x, \bar{b} \cdot axa^{-1} = a(\bar{b}x)a^{-1}$ eine weitere Instanz des Konjugationsproblems, wobei $x \notin Z(\bar{b} \cdot x)$ gilt. In $GL(2, \mathbb{Z}_p)$ reichen diese beiden Instanzen bereits aus, um ein Gruppenelement $\hat{a} = a \cdot z$ mit $z \in Z(GL(2, \mathbb{Z}_p))$ zu berechnen (siehe Abschnitt 2.7.4). Durch Konjugation mit $\hat{a}$ kann der Schlüssel nun effizient berechnet werden: $\hat{a}bxb^{-1}\hat{a}^{-1} = abxb^{-1}a^{-1}$.

Genau wie bei MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ haben die Instanzen des Konjugationsproblems, die bei diesen Protokollen auftreten, eine sehr spezielle Form. Die Elemente g, mit denen konjugiert wird, sind zwar selbst nicht bekannt. Aus den öffentlichen Parametern können aber effizient Elemente aus dem Zentralisator von g bestimmt werden. Mittels dieser Zentralisatorelemente kann eine weitere Instanz des Konjugationsproblems erzeugt werden, bei der mit dem selben Gruppenelement g konjugiert wurde. Dadurch erhält ein Angreifer zwei simultane Instanzen des CP in $GL(2, \mathbb{Z}_p)$, die er zur Lösung des speziellen Konjugationsproblems einsetzen und einen zum geheimen Schlüssel äquivalenten Wert berechnen kann.

[18] Wie bereits in Abschnitt 7.1 erwähnt wird dieses Problem in der jüngeren Literatur über Kryptografie auf Zopfgruppen auch als (Computational) Braid Diffie-Hellman Konjugationsproblem bezeichnet.

7.2.2 Der Cayley-Purser Algorithmus

Der Cayley-Purser Algorithmus [16] wurde im Jahre 1999 von der damals 16-jährigen irischen Schülerin Sarah Flannery vorgestellt, die mit diesem Projekt im Januar 1999 Siegerin der „ESAT Young Scientists and Technology Exhibition“ wurde. Sie benannte ihren Algorithmus nach Arthur Cayley (1821-1895), einem britischen Mathematiker, der bedeutende Beiträge zur Algebra von Matrizen leistete, und Michael Purser, der sie während eines Praktikums bei Baltimore Technologies zu ihrer Arbeit inspirierte und auf dessen Vorarbeit ihr Projekt aufbaute.
Sarah Flannery selbst fand eine grundlegende Schwäche in ihrer Arbeit, die sie als Anhang zusammen mit dem Algorithmus veröffentlichte [16]. Diese Schwäche besteht darin, dass als Teil des öffentlichen Schlüssels sowohl eine Instanz $a, b = x^{-1}a^{-1}x$ des Konjugationsproblems in $GL(2, \mathbb{Z}_n)$ als auch das Element $g = x^r$ veröffentlicht wird. Dadurch ist es möglich, das spezielle Konjugationsproblem in $GL(2, \mathbb{Z}_n)$ zu lösen, d.h. eine Matrix $\hat{x} \in GL(2, \mathbb{Z}_n)$ zu berechnen mit $Inn(\hat{x}) = Inn(x)$. Dieses Element $\hat{x}$ reicht bereits aus, um alle mit diesem öffentlichen Schlüssel verschlüsselten Nachrichten entschlüsseln zu können. Wir werden zunächst den Cayley-Purser Algorithmus kurz beschreiben und anschließend demonstrieren, wie dieser gebrochen werden kann.

Schlüsselerzeugung:

- Wähle zwei große Primzahlen p, q und berechne daraus den RSA-Modul $n = pq$.
- Wähle $x, a \in GL(2, \mathbb{Z}_n)$ mit $xa^{-1} \neq ax$ und berechne $b = x^{-1}a^{-1}x$.
- Wähle $r \in_R \mathbb{N}$ und berechne $g = x^r$.

Der Modul n sowie die Gruppenelemente $a, b, g \in GL(2, \mathbb{Z}_n)$ bilden den öffentlichen Schlüssel des Teilnehmers. Der zugehörige geheime Schlüssel besteht aus der Matrix x.

Verschlüsselung: Um eine Nachricht $M \in GL(2, \mathbb{Z}_n)$ bzgl. des öffentlichen Schlüssels (n, a, b, g) zu verschlüsseln, macht der Absender folgendes:

- Wähle $t \in_R \mathbb{N}$ und berechne $s = g^t$.
- Berechne $e = s^{-1}as$ und $k = s^{-1}bs$.
- Berechne $C = kMk$.

Das Paar (C, e) ist der Geheimtext von M.

Entschlüsselung: Um einen Geheimtext (C, e) zu entschlüsseln, berechnet der Empfänger zunächst $x^{-1}ex = x^{-1}(s^{-1}as)x = s^{-1}(x^{-1}ax)s = s^{-1}b^{-1}s = k^{-1}$. Die Klartext-Nachricht ergibt sich dann als $M = k^{-1}Ck^{-1}$.

Für die Entschlüsselung ist nicht unbedingt die Matrix x nötig. Eine Matrix $\hat{x} \in GL(2, \mathbb{Z}_n)$ mit $Inn(\hat{x}) = Inn(x)$ erfüllt den selben Zweck. Da mit $g = x^r$ ein Element aus dem Zentralisator $Z(x)$ von x gegeben ist, kann eine Variante des Angriffs auf das MOR System verwendet werden, um ein solches Element $\hat{x}$ zu berechnen:
Aus der Instanz $a^{-1}, b = x^{-1}a^{-1}x$ des Konjugationsproblems in $GL(2, \mathbb{Z}_n)$ erhält man durch Multiplikation mit dem öffentlichen Gruppenelement x^r eine weitere Instanz $x^r \cdot a^{-1}, x^r \cdot b = x^r x^{-1}a^{-1}x = x^{-1}(x^r \cdot a^{-1})x$ des Konjugationsproblems in $GL(2, \mathbb{Z}_n)$. Die simultane Lösung dieser beiden Instanzen liefert einen Wert $\hat{x} \in GL(2, \mathbb{Z}_n)$ mit $Inn(\hat{x}) = Inn(x^{-1})$ und damit einen zum geheimen Schlüssel äquivalenten Wert (siehe Abschnitt 2.7.4).

Die Tatsache, dass wir über $GL(2, \mathbb{Z}_n)$ mit einem zusammengesetzten Modul n anstatt $GL(2, \mathbb{Z}_p)$ mit primen Modul p rechnen ist dabei unwesentlich. Sollte eine der Rechnungen nicht ausführbar sein, hat man einen Teiler von n gefunden.
Der Algorithmus funktioniert auch, wenn man für n anstatt eines zusammengesetzten einen primen Modul wählt. Die Entscheidung, einen zusammengesetzten Modul zu verwenden, wird damit begründet, dass selbst wenn der Exponent r bekannt ist, der geheime Schlüssel x nicht aus dem Element g des öffentlichen Schlüssels berechnet werden kann, da die Faktorisierung von n nicht bekannt ist. Kann x so gewählt werden, dass das diskrete Logarithmus Problem schwer ist in $< x >$, so sollte der leichteren Handhabbarkeit wegen ein primer Modul verwendet werden.

7.2.3 Fazit

Wir wollen noch einmal die Gründe dafür zusammenfassen, dass das MOR System, die CDH-type CP basierten Verfahren und der Cayley-Purser Algorithmus unsicher sind auf den Gruppen $SL(2, \mathbb{Z}_p)$ bzw. $GL(2, \mathbb{Z}_p)$. Für die beschriebenen Angriffe und Schwächen wurden folgende Beobachtungen benutzt:

- **Das Konjugationsproblem ist leicht in $\mathbf{GL(2, \mathbb{Z}_p)}$**
  Sei x, wxw^{-1} eine Instanz des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$. Dann kann durch Lösen eines linearen Gleichungssystems über $\mathbb{Z}_p$ eine Lösung effizient berechnet werden. Die Lösungsmenge ist von der Form $L = w \cdot Z(x)$ (siehe Abschnitt 2.7.4).

- **Das spezielle Konjugationsproblem ist leicht in $\mathbf{GL(2, \mathbb{Z}_p)}$**
  Sei mit $Inn(g)$ eine Instanz des speziellen Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ gegeben. Wähle $x, y \in GL(2, \mathbb{Z}_p)$ mit $x \notin Z(y)$. Berechnet man die Bilder von x und y unter $Inn(g)$ erhält man mit x, gxg^{-1} und y, gyg^{-1} zwei Instanzen des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$. In Abschnitt 2.7.4 wurde gezeigt, dass eine simultane Lösung $h \in GL(2, \mathbb{Z}_p)$ dieser beiden Instanzen des Konjugationsproblems den selben inneren Automorphismus definiert wie g, d.h. eine Lösung der Instanz $Inn(g)$ des speziellen Konjugationsproblems in G darstellt.

Bei allen der diskutierten Verfahren hat ein Angreifer eine Instanz x, gxg^{-1} des Konjugationsproblems in $GL(2, \mathbb{Z}_p)$ zur Verfügung. Bei den CDH-type CP basierten Verfahren und dem Cayley-Purser Algorithmus sind diese als Teil des öffentlichen Schlüssels gegeben. Bei MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ müssen diese aus den Geheimtexten extrahiert werden. Um sein Ziel zu erreichen, muss ein Angreifer in allen Systemen ein Gruppenelement $\hat{g} \in GL(2, \mathbb{Z}_p)$ berechnen, dass sich nur um ein Zentrumselement von g unterscheidet, d.h. g und $\hat{g}$ müssen den selben inneren Automorphismus definieren.
Allein anhand der Instanz x, gxg^{-1} könnte ein Angreifer ein solches $\hat{g}$ nur durch Ausprobieren finden und bei richtiger Wahl der Parameter wäre seine Erfolgswahrscheinlichkeit vernachlässigbar klein. Der Angreifer kann jedoch zusätzlich ein Gruppenelement $h \in Z(g)$ berechnen, das es ihm ermöglicht, sein Ziel zu erreichen. Durch Multiplikation mit h erhält er mit $h \cdot x, h \cdot gxg^{-1} = h(h \cdot x)g^{-1}$ eine zweite Instanz des CP in $GL(2, \mathbb{Z}_p)$. Simultanes Lösen beider Instanzen liefert das gesuchte Element $\hat{g}$ (siehe Abschnitt 2.7.4).

Diese Beobachtungen fassen wir in folgendem Konstruktionsprinzip für kryptografische Verfahren auf $GL(2, \mathbb{Z}_p)$ zusammen, das in analoger Weise auch für andere endlich erzeugte nicht-abelschen Gruppen gilt, in denen das spezielle Konjugationsproblem durch eine simultane Lösung mehrerer Instanzen des Konjugationsproblems effizient lösbar ist:

> Wird ein geheimer Wert $g \in GL(2, \mathbb{Z}_p)$ zur Konjugation eines öffentlichen Wertes $x \in GL(2, \mathbb{Z}_p)$ benutzt und ist das Ergebnis gxg^{-1} ebenfalls öffentlich, so dürfen Elemente $\hat{g}$ aus dem Zentralisator $Z(g)$ von g nicht effizient berechnebar sein.

7.3 Entwicklung eines Commitment Schemes

Ziel des folgenden Abschnitts ist es, mittels der in Abschnitt 7.1 vorgestellten Bausteine und unter Vermeidung der in Abschnitt 7.2 identifizierten Designschwäche ein beweisbar sicheres Commitment Scheme auf der Gruppe $GL(2, \mathbb{Z}_p)$ zu konstruieren. Damit soll einerseits gezeigt werden, dass die Konstruktion sicherer kryptografischer Verfahren auf $GL(2, \mathbb{Z}_p)$ prinzipiell möglich ist, und zum anderen die praktische Anwendbarkeit der vorgestellten Bausteine demonstriert werden.
Ein Commitment Scheme ist ein interaktives Protokoll zwischen zwei Parteien, einem Sender und einem Empfänger. Es besteht aus 2 Phasen und lässt sich anschaulich anhand von verschließbaren Boxen erklären. In der ersten Phase, der Commitment Phase, legt der Sender eine Nachricht in eine solche Box, verschließt diese und sendet die Box samt Inhalt an den Empfänger. Am Ende der Commitment Phase ist eine Situation erreicht, in der zum einen der Empfänger nichts über den Inhalt der Box weiß, da er nicht im Besitz des passenden Schlüssels für diese Box ist (Secrecy-Eigenschaft). Zum anderen kann der Sender die Nachricht in der Box aber auch nicht mehr ändern, da sich die Box in der Obhut des Empfängers befindet (Binding-Eigenschaft). Zu diesem Zeitpunkt hat sich der Sender also auf den Inhalt der hinterlegten Nachricht festgelegt. Diesem Umstand hat das Commitment Scheme seinen Namen zu verdanken (Commitment = Verpflichtung, Bindung). In der zweiten, der Decommitment Phase, schickt der Sender den Schlüssel für die Box an den Empfänger und macht damit die hinterlegte Nachricht öffentlich.
Eine Anwendung von Commitment Schemes sind öffentliche Ausschreibungen, bei denen jeder Bewerber innerhalb einer vorher festgesetzten Frist ein geheimes Angebot abgeben muss. Die Angebote werden in versiegelten Umschlägen übergeben, die bei der ausschreibenden Stelle gesammelt und erst nach Ablauf der Frist geöffnet werden. Dadurch ist gewährleistet, dass die Bewerber während der Frist ein bindendes Angebot abgeben können, ohne dass dieses den Mitbewerbern bekannt wird.

Bei der folgenden formalen Definition steht Algorithmus $\mathcal{S}$ für den Sender und Algorithmus $\mathcal{R}$ für den Empfänger (Receiver). Die zusätzlichen Eingaben *comm* und *decom* für den Sender $\mathcal{S}$ dienen zur Unterscheidung der beiden Phasen des Commitment Schemes. Alternativ könnte man auch zwei Algorithmen S_1 und S_2 definieren, die den Sender in der Commitment bzw. in der Decommitment Phase abbilden.

Definition 28 (Commitment Scheme) *Ein Paar $(\mathcal{S}, \mathcal{R})$ bestehend aus einem probabilistischen Algorithmen $\mathcal{S}$ mit erwartet polynomieller Laufzeit und einem deterministischen Algorithmus $\mathcal{R}$ mit polynomieller Laufzeit, der entweder akzeptiert oder verwirft, heißt $\mathcal{M}$-Commitment Scheme, wenn es folgende Eigenschaft erfüllt:*

- *Completeness-Eigenschaft: Für alle Nachrichten $m \in \mathcal{M}$ gilt*

$$\mathcal{R}(1^n, v, m, \mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}}), \mathcal{S}(decom, 1^n, v, m, r_{\mathcal{S}})) = accept.$$

Dabei bezeichnet $r_{\mathcal{S}}$ die Zufallsbits von Algorithmus $\mathcal{S}$ und v öffentliche Parameter, die beiden Teilnehmern bekannt sind (etwa die benutzte Gruppe) und n den Sicherheitsparameter. Ein Commitment Scheme heißt sicher, wenn zusätzlich gilt:

- *Secrecy-Eigenschaft: Für alle probabilistischen Algorithmen $\mathcal{A}$ mit erwartet polynomieller Laufzeit und alle Polynome q gilt*

$$\Pr[\mathcal{A}(1^n, v, \mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}})) = m] < \frac{1}{q(n)}.$$

- *Unambiguity-Eigenschaft: Für alle probabilistischen Algorithmen $\mathcal{B}$ mit erwartet polynomieller Laufzeit und alle Polynome q gilt für $(m, \hat{m}, C, D, \hat{D}) = \mathcal{B}(1^n, v)$ mit $m \neq \hat{m}$*

$$\Pr[\mathcal{R}(1^n, v, m, C, D) = \mathcal{R}(1^n, v, \hat{m}, C, \hat{D}) = accept] < \frac{1}{q(n)}.$$

Der Wert $\mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}})$ stellt das eigentliche Commitment von $\mathcal{S}$ zur Nachricht $m \in \mathcal{M}$ dar. In der Decommitment Phase öffnet $\mathcal{S}$ sein Commitment, indem er die hinterlegte Nachricht m und eine Art „Nachweis“ $\mathcal{S}(decom, 1^n, v, m, r_{\mathcal{S}})$ sendet, anhand dessen der Empfänger die Korrektheit der Öffnung überprüfen kann. Dieser „Nachweis“ kann zum Beispiel aus den für die Berechnung des Commitments $\mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}})$ benutzten kryptografischen Sitzungsschlüsseln und den verwendeten Zufallszahlen bestehen. Je nach Ergebnis dieser Überprüfung wird der Empfänger entweder akzeptieren (Ausgabe *accept*) oder verwerfen (Ausgabe $reject$). Die Completeness-Eigenschaft stellt sicher, dass der Empfänger immer akzeptiert, sofern sich Sender und Empfänger an das vorgegebene Protokoll halten.
Die Secrecy-Eigenschaft stellt sicher, dass in einem sicheren Commitment Scheme aus der Ausgabe $\mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}})$ des Senders nicht die hinterlegte Nachricht m berechnet werden kann. Die Definition lässt allerdings zu, dass es Algorithmen geben kann, die gewisse Eigenschaften, z.B. das unterwertigste Bit, der Nachricht m berechnen. Bei Existenz eines solchen Algorithmus würde das betrachtete Commitment Scheme immer noch als sicher im Sinne der obigen Definition gelten. Solche Situationen kann man durch eine Verschärfung der Definition der Secrecy-Eigenschaft ausschließen. Ähnlich wie bei der Definition der Ununterscheidbarkeit für Public-Key Verschlüsselungen fordert man, dass es keinen Algorithmus gibt, der bei Kenntnis von zuvor selbstgewählten Nachrichten $m_0, m_1 \in \mathcal{M}$ und einem Commitment $C(m_i)$ zu m_i für ein zufälliges $i \in \{0, 1\}$ das Bit i mit einer Wahrscheinlichkeit bestimmen kann, die nicht-vernachlässigbar größer als 0,5 ist.
In obiger Definition wird ferner vorausgesetzt, dass die Commitment Phase lediglich aus der Berechnung von $\mathcal{S}(comm, 1^n, v, m, r_{\mathcal{S}})$ und der Übertragung dieses Wertes an den Empfänger $\mathcal{R}$ besteht. Lässt man für die Commitment Phase auch interaktive Protokolle zwischen Sender und Empfänger zu, so muss man für $\mathcal{R}$ auch probabilistische Algorithmen zulassen und die Ergebnisse $\mathcal{S}(comm, m, r_{\mathcal{S}})$ der Berechnungen von $\mathcal{S}$ in der Commitment Phase durch die Sicht (View) des Empfängers in dieser Phase ersetzen.

Für die Algorithmen $\mathcal{A}$ bzw. $\mathcal{B}$ in der Definition der Secrecy- bzw. Unambiguity-Eigenschaft haben wir polynomiell (im Sicherheitsparameter n) beschränkte Laufzeiten gefordert. Lässt man auch Algorithmen mit unbeschränkter Laufzeit zu, so wäre die entsprechende Eigenschaft informationstheoretisch geschützt, d.h. selbst ein Angreifer mit unbeschränkter Rechenleistung könnte die hinterlegte Nachricht m nicht berechnen bzw. wäre nicht in der Lage, ein Commitment auf zwei verschiedene Arten zu öffnen. Es kann allerdings kein Commitment Scheme geben, bei dem beide Eigenschaften informationstheoretisch geschützt sind. Bei mindestens einer der Sicherheitseigenschaften muss man sich auf Angreifer mit polynomieller Laufzeit beschränken.

7.3.1 Konstruktion eines beweisbar sicheren Commitment Schemes auf nicht-abelschen Gruppen

Wir werden jetzt die in den letzten Abschnitten definierten Probleme und Einwegfunktionen benutzen, um ein Commitment Scheme für nicht-abelsche Gruppen zu entwickeln und eine konkrete Umsetzung auf der Gruppe $GL(2, \mathbb{Z}_p)$ angeben.
Grundlage für dieses Protokoll soll wieder eine Kombination aus dem Diskreter Logarithmus Problem und dem Konjugationsproblem sein. Die hinterlegte Nachricht soll dabei in den Exponenten kodiert werden. Wir überlegen uns zunächst, dass das DLCP (siehe Abschnitt 7.1.1) und die Einwegfunktion $F_{g,h}$ aus Abschnitt 7.1.3 nur bedingt geeignet sind für ein Commitment Scheme.
Benutzt man das DLCP als Commitment Scheme, so bestünde ein Commitment zu m aus dem Tupel $(g, h, g^m h g^{-m})$. Dies hat zum einen den Nachteil, dass dieses Commitment Scheme deterministisch ist. Ein Angreifer könnte insbesondere vergleichen, ob in zwei vorliegenden Commitments die selben Nachrichten hinterlegt wurden. Zum anderen kann ein Angreifer aus einem Commitment $(g, h, g^m h g^{-m})$ zur Nachricht m leicht ein Commitment $(g, h, g^k g^m h g^{-m} g^{-k})$ zur Nachricht $m + k$ berechnen.
Grundsätzlich können injektive Einwegfunktionen zur Konstruktion von Commitment Schemes benutzt werden. Als Commitment zu einer Nachricht m dient dabei das Bild von m unter der Einwegfunktion. Zum Öffnen des Commitments veröffentlicht der Sender die hinterlegte Nachricht. Der Empfänger kann durch Berechnen des Bildes von m überprüfen, ob das Commitment korrekt geöffnet wurde.
In Abschnitt 7.1.3 wurde die Einwegfunktion $F_{g,h}$ definiert, die auf einer Kombination aus DLP und CP basiert. Die Nachricht könnte als Exponent von g oder h hinterlegt werden. Im ersten Fall hätte ein Commitment die Form $C(m) = g^m h^r g^{-m}$ für ein zufälliges $r \in_R \mathbb{Z}_q$ und hätte ähnliche Nachteile wie die Benutzung des DLCP. Im zweiten Fall hätte ein Commitment die Form $C(m) = g^r h^m g^{-r}$ für ein zufälliges $m \in_R \mathbb{Z}_p$. Diese Variante eignet sich nur bedingt für die Umsetzung auf Matrixgruppen. Da ein Element $g \in G$ bekannt ist, leidet das Commitment Scheme in diesem Fall unter den in Abschnitt 7.2.3 aufgedeckten Designfehlern, die die Berechnung eines Elements $\hat{g} \in G$ mit $Inn(\hat{g}) = Inn(g^r)$ ermöglichen, wenn das CP in G leicht ist. Die Sicherheit des Verfahrens würde in diesem Fall auf der Schwierigkeit des Diskreter Logarithmus Problems in der Gruppe $< h >$ beruhen. Da ein Angreifer den Wert h^m aus dem Commitment $C(m)$ berechnen kann, könnte er wieder

Commitments zur Nachricht $l \cdot m + k \pmod q$ für beliebige $k, l \in \mathbb{Z}_q$ berechnen. Um solche elementaren Schwächen zu verhindern, benutzen wir nicht die Einwegfunktion $F_{g,h}$ direkt, sondern wandeln diese leicht ab.

Die modifizierte Einwegfunktion: Sei G eine nicht-abelsche Gruppe und seien $g, h, i \in G$ mit $i, g \notin Z(h)$ und $\text{ord}(h) = p$ mit p prim. Dann definieren wir die Funktion $F_{g,h,i}$ wie folgt:

$$\begin{aligned} F_{g,h,i} : \mathbb{Z}_s \times \mathbb{Z}_p &\rightarrow G \\ (x, y) &\mapsto g^x h^y i h^{-y} g^{-x} \end{aligned}$$

Durch die Forderung, dass $i \notin Z(h)$ und dass $\text{ord}(h) = p$ mit p prim ist, wird sichergestellt, dass $h^k i h^{-k} \neq i$ für alle $k \not\equiv 0 \pmod p$ gilt. Die Forderung $g \notin Z(h)$ wurde schon im Hinblick auf den späteren Einsatz der Funktion $F_{g,h,i}$ in einem Commitment Scheme aufgenommen. Der hinterlegte Wert wird im Exponenten von h stehen. Wäre $g \in Z(h)$, so könnte ein Angreifer den Wert des Commitments ändern, indem er es mit h^k für ein $k \in \mathbb{Z}$ konjugiert. Da h prime Ordnung hat, gilt $h^k \notin Z(g)$ für alle $k \not\equiv 0 \pmod p$ und dieser einfache Angriff funktioniert nicht.

Satz 36 *Wenn das DLP in $<h>$ schwer ist, dann ist y ein Hard-Core Prädikat der Funktion $F_{g,h,i}$.*

Beweis: Ist mit h, h^y eine Instanz des DLP in $<h>$ gegeben, so wählt man ein zufälliges $x \in_R \mathbb{Z}_p$ und berechnet den Funktionswert $F_{g,h,i}(x, y) = g^x h^y i h^{-y} g^{-x}$. Existiert ein Algorithmus $\mathcal{A}$, der y berechnet, so liefert $\mathcal{A}$ auf Eingabe $g^x h^y i h^{-y} g^{-x}$ den gesuchten diskreten Logarithmus. □

Diese Reduktion setzt voraus, dass die betrachtete Funktion $F_{g,h,i}$ injektiv ist. Gibt es Urbilder $(x_1, y_1), \ldots, (x_n, y_n)$ mit $F_{g,h,i}(x_i, y_i) = F_{g,h,j}(x_i, y_j)$ für $1 \leq i, j \leq n$, so liefert $\mathcal{A}$ eines der y_i. Das ausgegebene y_i muss aber nicht der gesuchte diskrete Logarithmus sein. Liefert der Algorithmus $\mathcal{A}$ ein zufälliges y_i, so genügt es zu fordern, dass jeweils nur maximal polynomiell viele (im benutzten Sicherheitsparameter) Urbilder auf ein Bild abgebildet werden. Die Reduktion funktioniert in diesem Falle auch dann, wenn $F_{g,h,i}$ nicht injektiv ist. Der folgende Satz gibt eine Bedingung an die Wahl der Parameter für die Injektivität der Funktion $F_{g,h,i}$ an.

Satz 37 *Die Funktion $F_{g,h,i}$ ist genau dann injektiv, wenn für alle $r, s, t \in \mathbb{Z}$ mit $r, t \not\equiv 0 \pmod p$ und $s \not\equiv 0 \pmod{\text{ord}(g)}$ gilt, dass $h^r g^s h^t \notin Z(i)$ erfüllt ist.*

Beweis: Wir nehmen an, dass es $r, s, t \in \mathbb{Z}$ mit $r, t \not\equiv 0 \pmod p$ und $s \not\equiv 0 \pmod{\text{ord}(g)}$ gibt, so dass $h^r g^s h^t \in Z(i)$, d.h. $h^r g^s h^t i = i h^r g^s h^t$. Schreibt man $s = s_1 + s_2$ als Summe mit $s_1, s_2 \not\equiv 0 \pmod{\text{ord}(g)}$ und $s_1 \not\equiv -s_2 \pmod{\text{ord}(g)}$, so lässt sich diese Gleichung äquivalent umformen zu $g^{s_1} h^t i h^{-t} g^{-s_1} = g^{-s_2} h^{-r} i h^r g^{s_2}$. In diesem Fall gilt also $F_{g,h,i}(s_1, t) = F_{g,h,i}(-s_2, -r)$. □

Das resultierende Commitment Scheme:
Wir werden nun ein Commitment Scheme auf nicht-abelschen Gruppen angeben, bei dem das Commitment $c \in G$ zu einer Nachricht $m \in \mathbb{Z}_p$ aus dem Funktionswert $F_{g,h,i}(r,m)$ besteht, wobei $r \in_R \mathbb{Z}_s$ zufällig gewählt ist. Das Öffnen des Commitments besteht aus der Offenlegung des zum Commitment $F_{g,h,i}(r,m)$ gehörenden Urbildes (r,m). Ist $F_{g,h,i}$ injektiv, so kann der Sender sein Commitment auf genau eine Weise öffnen (Unambiguity-Eigenschaft). Die Secrecy-Eigenschaft folgt mit Satz 36, wenn die benutzte Gruppe G und der öffentliche Parameter $h \in G$ so gewählt wurden, dass das diskrete Logarithmus Problem schwer ist in $<h>$.

Wahl der Parameter: Sei G eine nicht-abelsche Gruppe. Wähle $g,h,i \in G$ mit $\text{ord}(h) = p$, $\text{ord}(g) = q$, wobei p,q prim, und $i,g \notin Z(h)$.
Die Werte $g,h,i \in G$ müssen ferner so gewählt werden, dass die Funktion $F_{g,h,i}$ injektiv ist.

Commitment Phase: Wähle $r \in_R \mathbb{Z}_p$ zufällig. Das Commitment zu $m \in \mathbb{Z}_q$ besteht aus dem Wert $C(m) = F_{g,h,i}(r,m) = g^r h^m i h^{-m} g^{-r}$.

Decommitment Phase: Zum Öffnen von $C(m,r)$ legt der Sender die Werte m und r vor. Der Empfänger akzeptiert genau dann, wenn der von ihm berechnete Funktionswert $F_{g,h,i}(r,m)$ mit dem Wert, den er in der Commitment Phase erhalten hat, übereinstimmt.

Halten sich Sender und Empfänger an das Protokoll, so wird der Empfänger immer akzeptieren. Das Protokoll erfüllt also die Completeness-Eigenschaft. Die Unambiguity-Eigenschaft folgt direkt aus der Injektivität der benutzten Funktion $F_{g,h,i}$. Mit Satz 36 folgt, dass die Secrecy-Eigenschaft erfüllt ist, wenn das DLP schwer ist in der Gruppe $<h>$.

7.3.2 Umsetzung auf $GL(2,\mathbb{Z}_p)$

Seien p,q prim mit $q \mid p-1$ und sei G_q die eindeutige bestimmte zyklische Untergruppe von $\mathbb{Z}_p^*$ der Ordnung q.

Wahl der Parameter: Wähle $a_1,a_2,a_4,b_1,b_2,b_4 \in_R G_q$ mit $a_1 \neq a_4$, $b_1 \neq b_4$ und $a_1b_2 + a_2b_4 \not\equiv a_2b_1 + a_4b_2 \pmod p$. Dann haben die Matrizen $A = \begin{pmatrix} a_1 & a_2 \\ 0 & a_4 \end{pmatrix}$, $B = \begin{pmatrix} b_1 & b_2 \\ 0 & b_4 \end{pmatrix} \in GL(2,\mathbb{Z}_p)$ Ordnung q (siehe Satz 8 auf Seite 27). Die Bedingung $a_1b_2 + a_2b_4 \not\equiv a_2b_1 + a_4b_2 \pmod p$ stellt sicher, dass $A \notin Z(B)$ gilt. Da A und B beide Ordnung q haben, folgt

$$B^l \notin Z(A^k) \quad \text{für } k,l \not\equiv 0 \pmod q,$$

d.h. kein Element aus der von A erzeugten Gruppe (außer der Einheitsmatrix) kommutiert mit einem Element aus der von B erzeugten Gruppe (außer der Einheitsmatrix).

Wähle $C = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} \in GL(2, \mathbb{Z}_p)$ mit $c_3 \not\equiv 0 \pmod{p}$. Dann gilt $C \notin Z(B)$ und wegen der primen Ordnung von B folgt

$$C \notin Z(B^k) \quad \text{für } k \not\equiv 0 \pmod{p}.$$

Commitment Phase: Wähle $r \in_R \mathbb{Z}_q$ zufällig. Das Commitment zu $m \in \mathbb{Z}_q$ ist $F_{A,B,C}(r, m) = A^r B^m C B^{-m} A^{-r}$.

Decommitment Phase: Zum Öffnen des Commitments legt der Sender die Werte m und r vor. Der Empfänger akzeptiert genau dann, wenn der von ihm berechnete Funktionswert $F_{A,B,C}(r, m)$ mit dem Wert, den er in der Commitment Phase erhalten hat, übereinstimmt.

Halten sich Sender und Empfänger an das Protokoll, so wird der Empfänger immer akzeptieren. Das Protokoll erfüllt also die Completeness-Eigenschaft. Um die Secrecy- und die Unambiguity-Eigenschaft zu beweisen, brauchen wir, dass die Funktion $F_{A,B,C}$ injektiv ist.

Satz 38 *Seien $A, B, C \in GL(2, \mathbb{Z}_p)$ so gewählt, wie unter „Wahl der Parameter“ beschrieben. Dann ist die Funktion $F_{A,B,C}$ injektiv.*

Beweis: Angenommen, es gibt $x, y, \hat{x}, \hat{y} \in \mathbb{Z}_q$, für die $A^x B^y C B^{-y} A^{-x} = F_{A,B,C}(x, y) = F_{A,B,C}(\hat{x}, \hat{y}) = A^{\hat{x}} B^{\hat{y}} C B^{-\hat{y}} A^{-\hat{x}}$ gilt. Dies ist genau dann der Fall, wenn $B^{-\hat{y}} A^{x-\hat{x}} B^y C = C B^{-\hat{y}} A^{x-\hat{x}} B^y$, d.h. $B^{-\hat{y}} A^{x-\hat{x}} B^y \in Z(C)$ gilt. Die Matrix $B^{-\hat{y}} A^{x-\hat{x}} B^y$ ist als Produkt von oberen Dreiecksmatrizen selbst wieder eine obere Dreiecksmatrix. Da $C \in GL(2, \mathbb{Z}_p)$ mit $(2,1)$-Komponente $c_3 \not\equiv 0 \pmod{p}$ gewählt wurde, muss $B^{-\hat{y}} A^{x-\hat{x}} B^y = c \cdot I$ für ein $c \in \mathbb{Z}_p$ sein (siehe auch Abschnitt 2.7.3 auf Seite 28 über Zentralisatoren in linearen Gruppen). Damit gilt $B^{-\hat{y}} A^{x-\hat{x}} B^y \in Z(GL(2, \mathbb{Z}_p))$ und es folgt

$$\begin{aligned} B^{-\hat{y}} A^{x-\hat{x}} B^y &= Inn(B^y)(B^{-\hat{y}} A^{x-\hat{x}} B^y) = B^{y-\hat{y}} A^{x-\hat{x}} \quad \text{und} \\ B^{-\hat{y}} A^{x-\hat{x}} B^y &= Inn(B^{\hat{y}})(B^{-\hat{y}} A^{x-\hat{x}} B^y) = A^{x-\hat{x}} B^{y-\hat{y}}. \end{aligned}$$

Insgesamt folgt, dass $B^{y-\hat{y}} A^{x-\hat{x}} = A^{x-\hat{x}} B^{y-\hat{y}}$ und damit $A^{x-\hat{x}} \in Z(B^{y-\hat{y}})$ gilt.
Da $\text{ord}(A) = \text{ord}(B) = q$ mit q prim und $A \notin Z(B)$ ist, folgt $x - \hat{x} \equiv 0 \pmod{q}$ oder $y - \hat{y} \equiv 0 \pmod{q}$. Im Falle $x - \hat{x} \equiv 0 \pmod{q}$ wäre $B^{y-\hat{y}} = c \cdot I$ und damit $b_1^{y-\hat{y}} = b_4^{y-\hat{y}}$. Angenommen $y - \hat{y} \not\equiv 0 \pmod{q}$. Sei $k \equiv (y - \hat{y})^{-1} \pmod{q}$. Dann folgt $(b_1^{y-\hat{y}})^k = b_1 = b_4 = (b_4^{y-\hat{y}})^k$, was ein Widerspruch zur Wahl von $B \in GL(2, \mathbb{Z}_p)$ ist. Also gilt $y - \hat{y} \equiv 0 \pmod{q}$. Analog lässt sich zeigen, dass im Fall $y - \hat{y} \equiv 0 \pmod{q}$ auch $x - \hat{x} \equiv 0 \pmod{q}$ gelten muss. Insgesamt folgt $x - \hat{x} \equiv 0 \pmod{q}$ und $y - \hat{y} \equiv 0 \pmod{q}$ und damit die Injektivität von $F_{A,B,C}$. □

Satz 39 *Das vorgestellte Commitment Scheme auf $GL(2, \mathbb{Z}_p)$ ist sicher unter der Annahme, dass das Diskreter Logarithmus Problem schwer ist in G_q.*

Beweis: Zu zeigen sind die Secrecy- und die Unambiguity-Eigenschaft. Die Unambiguity-Eigenschaft folgt direkt aus der Injektivität der Funktion $F_{A,B,C}$ (sogar gegen unbeschränkte Angreifer). Wir zeigen nun, dass ein Angreifer $\mathcal{A}$, der aus einem gegebenen Commitment $F_{A,B,C}(r,m) = A^r B^m C B^{-m} A^{-r}$ die hinterlegte Nachricht m berechnen kann, auch diskrete Logarithmen in der Gruppe $< B >$ berechnen kann: Sei mit B, B^x eine Instanz des DLP in $< B >$ gegeben. Wähle $r \in_R \mathbb{Z}_p$ zufällig, berechne $F_{A,B,C}(r,x) = A^r B^x C B^{-x} A^{-r}$, gib diesen Wert als Eingabe an $\mathcal{A}$ und mache die selbe Ausgabe wie $\mathcal{A}$. Die Erfolgswahrscheinlichkeit dieses Algorithmus zur Berechnung diskreter Logarithmen in $< B >$ ist gleich der Erfolgswahrscheinlichkeit von $\mathcal{A}$.
Der Parameter $B \in GL(2, \mathbb{Z}_p)$ wurde so gewählt, dass das DLP in $< B >$ genauso schwer ist wie das DLP in G_q (siehe auch Satz 8 auf Seite 27). Damit folgt die Behauptung. □

8 Zusammenfassung

Die vorliegende Arbeit gliedert sich inhaltlich in zwei Teile. Im ersten Teil wird das im Jahre 2001 von Paeng, Ha, Kim, Chee und Park vorgeschlagene MOR Kryptosystem [42] behandelt. Der zweite Teil greift die Idee auf, das Diskreter Logarithmus Problem und das Konjugationsproblem zu verknüpfen, um neue Probleme zu erhalten, die zur Konstruktion kryptografischer Verfahren auf nicht-abelschen Gruppen verwendet werden können.

Das im ersten Teil behandelte MOR Kryptosystem ist ein asymmetrisches Verschlüsselungsverfahren, das auf speziellen nicht-abelschen Gruppen ausgeführt werden kann. Zuerst wird mittels des Diffie-Hellman Verfahrens (siehe Abschnitt 2.4) ein Sitzungsschlüssel in der Gruppe der inneren Automorphismen $Inn(G)$ vereinbart, der anschließend zur Verschlüsselung der Nachricht verwendet wird. Das MOR System ist daher dem in Abschnitt 2.5 vorgestellten ElGamal-Verfahren sehr ähnlich.
Eine notwendige Bedingung für die Sicherheit des MOR Systems ist die Schwierigkeit der Berechnung diskreter Logarithmen in der Gruppe $Inn(G)$. Darüber hinaus muss die benutzte nicht-abelsche Gruppe G aber noch weitere Anforderungen erfüllen: Die Gruppe G muss endlich erzeugt sein, das Wortproblem muss in G effizient lösbar sein, das Zentrum der Gruppe darf weder zu klein noch zu groß sein und G muss effizient implementierbar sein.
Mit $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ [42] und $GL(2,R) \times_\theta \mathbb{Z}_n$ [43] wurden bislang zwei Gruppen vorgeschlagen, die diese Bedingungen erfüllen. In den Kapiteln 4 und 5 wird die Sicherheit des MOR Systems bei Benutzung dieser Gruppen eingehend analysiert. In Kapitel 4 zeigen wir, wie in MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ bei Verwendung der in [42] vorgeschlagenen Parameter der geheime Schlüssel effizient aus dem öffentlichen Schlüssel berechnet werden kann. Wir geben eine verbesserte Erzeugung des Parameter an, die MOR resistent gegen diesen Angriff macht. Es werden weitere Angriffe vorgestellt, deren Erfolg vom verwendeten Betriebsmodus von MOR abhängt. Für die Grundversion wird ein Ciphertext-Only Angriff angegeben, der für einen gegebenen Geheimtext die Menge der möglichen Klartexte auf einen (im Sicherheitsparameter) exponentiell kleinen Teil des Klartextraumes einschränkt. Insbesondere kann die Klartext-Matrix komplett berechnet werden, wenn eine Komponente von ihr bekannt ist.
Für die effizienten Modi von MOR auf $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$, die Berechnungen in $Inn(G)$ für mehrere Ver- und Entschlüsselungen benutzen, wurde ein Known-Plaintext Angriff entwickelt, der bei Kenntnis von zwei Paaren bestehend aus Klar- und zugehörigem Geheimtext die Berechnung eines zum geheimen Schlüssel äquivalenten Wertes ermöglicht. Die Angriffe funktionieren selbst bei Benutzung des in [42] vorgeschlagenen probabilistischen Padding-Verfahrens.
Von den insgesamt 8 Betriebsmodi von MOR erweist sich lediglich einer als resistent gegen die vorgestellten Angriffe. Da Ver- und Entschlüsselung in diesem Modus ca. 2550-mal bzw. ca. 32-mal langsamer sind als bei Verwendung von RSA (siehe Abschnitt 4.1), ist dieser Modus für den praktischen Einsatz jedoch zu ineffizient.

In [43] wird die Konstruktion einer Gruppe $GL(2,R) \times_\theta \mathbb{Z}_n$ basierend auf einem gegebenen Ringautomorphismus $\Phi : R \to R$ vorgestellt. Die Sicherheit von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ hängt wesentlich von der Wahl von Φ ab. In Kapitel 5 wird gezeigt, dass Teile des geheimen Schlüssels effizient aus den öffentlichen Parametern berechnet werden können, sofern die Berechnung diskreter Logarithmen effizient in $<\Phi>$ möglich ist. Die effizienten Betriebsmodi von MOR auf $GL(2,R) \times_\theta \mathbb{Z}_n$ haben sich sogar als unsicher gegen Chosen-Ciphertext Angriffe herausgestellt, wenn das schwächere Computational Diffie-Hellman Problem in $<\Phi>$ leicht ist. Bei den vorgestellten Angriffen handelt es sich um generische Angriffe, die in jedem kommutativen Ring mit Eins durchführbar sind. Bei Benutzung eines speziellen Rings kann es sogar noch stärkere Angriffe geben.

Aufgrund der aufgedeckten Sicherheitslücken eignen sich sowohl $SL(2,\mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ als auch $GL(2,R) \times_\theta \mathbb{Z}_n$ nicht für eine Benutzung im MOR System. Derzeit ist keine Gruppe bekannt, auf der das MOR System sicher und effizient ausgeführt werden kann. Die Suche nach Gruppen für das MOR System ist die derzeit dominierende offene Fragestellung im Umfeld des MOR Systems. In Kapitel 6 untersuchen wir die Nutzbarkeit von p-Gruppen für das MOR System. Nicht-abelsche p-Gruppen G_p erfüllen die notwendigen Bedingungen an die Größe des Zentrums. Wir geben eine Darstellung von p-Gruppen an, die ferner eine effiziente Implementierbarkeit und eine effiziente Lösung des Wortproblems gestattet. Unter der Annahme, dass das DLP in $Inn(G_p)$ schwer ist, erfüllen p-Gruppen damit alle notwendigen Bedingungen für eine Benutzung im MOR System. Ob die Eigenschaften von p-Gruppen unter der Annahme, dass das DLP schwer in $Inn(G_p)$ ist, bereits hinreichend für die Sicherheit von MOR sind, ist ein weiteres offenes Problem.

Der zweite Teil der Arbeit untersucht die Möglichkeit, durch Kombination klassischer kryptografischer Probleme mit dem Konjugationsproblem neue Probleme auf nicht-abelschen Gruppen zu konstruieren und diese für die Konstruktion kryptografischer Verfahren auf nicht-abelschen Gruppen zu verwenden. Dieser Teil ist motiviert durch das der Sicherheit des MOR Systems zugrundeliegende DLP in $Inn(G)$, das eng mit DLP und SCP in der zugrundeliegenden Gruppe G verwandt ist, und den kryptografischen Protokollen auf Zopfgruppen (siehe Abschnitt 7.2.1), deren Sicherheit auf solchen Kombinationen basiert.

Zunächst werden mehrere sinnvolle Kombinationen des Konjugationsproblems mit dem Diskreter Logarithmus Problem und den Diffie-Hellman Problemen vorgestellt und die Abhängigkeiten der neuen Probleme untereinander und zu ihren Bausteinen untersucht. Von zentraler Bedeutung für ihre Nutzbarkeit in kryptografischen Anwendungen ist dabei die Schwierigkeit dieser Probleme. In vielen Fällen lässt sich die Lösbarkeit eines Teilproblems auf die Lösbarkeit des kombinierten Problems reduzieren. In diesen Fällen ist das kombinierte Problem also mindestens so schwer wie das entsprechende Teilproblem. Auf diese Weise erhält man auch für nicht-abelsche Gruppen Probleme und Annahmen, die mindestens so stark sind wie die entsprechenden Probleme und Annahmen in der klassischen Kryptografie und sich für die Konstruktion kryptografischer Verfahren auf nicht-abelschen Gruppen nutzen lassen. Für die Gruppe $GL(2,\mathbb{Z}_p)$, die häufig für die Konstruktion krypto-

grafischer Protokolle verwendet wird, konnten wir zeigen, dass die neuen Kombinationsprobleme äquivalent zum Diskreter Logarithmus Problem bzw. den Diffie-Hellman Problemen in $GL(2, \mathbb{Z}_p)$ sind. Die Untersuchung der Gruppe $GL(2, \mathbb{Z}_p)$ ist auch aus dem Grund interessant, da sich die für die Gruppe $GL(2, \mathbb{Z}_p)$ erzielten Ergebnisse auf andere endlich erzeugte nicht-abelschen Gruppen übertragen lassen, in denen simultane Instanzen des Konjugationsproblems und damit das spezielle Konjugationsproblem effizient lösbar sind.

Mit dem Cayley-Purser Algorithmus [16] und den in [23] vorgestellten Algorithmen auf Zopfgruppen existieren bereits kryptografische Verfahren, deren Sicherheit auf der Schwierigkeit der in diesem Abschnitt diskutierten Kombinationsprobleme beruht. Die zunächst für Zopfgruppen vorgeschlagenen Protokolle aus [23] können auch auf anderen nicht-abelschen Gruppen ausgeführt werden. Eine Umsetzung dieser Protokolle auf $GL(2, \mathbb{Z}_p)$ hat sich als unsicher herausgestellt. Wir zeigen, dass eine Umsetzung der Protokolle aus [23] auf $GL(2, \mathbb{Z}_p)$, MOR auf $SL(2, \mathbb{Z}_p) \times_\theta \mathbb{Z}_p$ und der Cayley-Purser Algorithmus, der die Gruppe $GL(2, \mathbb{Z}_n)$ benutzt, anfällig für die selbe Art von Angriffen sind. Im Falle des Cayley-Purser Algorithmus und der Protokolle aus [23] können diese Angriffe benutzt werden, um einen zum geheimen Schlüssel äquivalenten Wert zu berechnen.
Wir geben ein Designprinzip für kryptografische Protokolle auf der Gruppe $GL(2, \mathbb{Z}_p)$ an, das die vorgestellten Angriffe unwirksam werden lässt.
Die vorgestellten Probleme, die wir auch zur Konstruktion von Einwegfunktionen verwenden, können als Bausteine zum Design kryptografischer Verfahren auf nicht-abelschen Gruppen verwendet werden. Die Nutzbarkeit dieser Bausteine verdeutlichen wir beispielhaft durch Konstruktion eines effizienten, beweisbar sicheren Commitment Schemes. Wir stellen das Commitment Scheme zunächst in einer allgemeinen Version für nicht-abelsche Gruppen vor, die gewisse Eigenschaften erfüllen. Das vorgestellte Protokoll vermeidet die bei den zuvor untersuchten Protokollen festgestellten Designfehler und lässt sich auch sicher auf der Gruppe $GL(2, \mathbb{Z}_p)$ umsetzen.

Literatur

[1] I. Anshel, M. Anshel, B. Fisher D. Goldfeld, „New key agreement protocols in braid group cryptography“, Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, LNCS 2020, S. 1-15

[2] I. Anshel, M. Anshel, D. Goldfeld, „An Algebraic Method for Public-Key Cryptography“, Mathematical Research Letters, 6 (1999), S. 287-291

[3] M. Baumgart, „Untersuchung der Gruppen $GL(s, \mathbb{Z}_p)$ und $SL(s, \mathbb{Z}_p)$ zur Nutzung in der Kryptographie“, Doktorarbeit, Justus-Liebig-Universität Gießen, erscheint wahrscheinlich 2004

[4] A. Beutelspacher, J. Schwenk, K.-D. Wolfenstetter, „Moderne Verfahren der Kryptographie“, Vieweg-Verlag, 1995

[5] S. Blackburn, S. Galbraith, „Cryptanalysis of two cryptosystems based on group action“, Advances in Cryptology - Asiacrypt 1999, LNCS 1716, S. 52-61

[6] B. den Boer, „Diffie-Hellman is as strong as discrete log for certain primes“, Advances in Cryptology - Crypto 1988, LNCS 403, S. 530-539

[7] M. Campagna, „Algorithms in Braid Groups“, IACR EPrint-Server, Report 2003/099, http://eprint.iacr.org/2003/099

[8] J. C. Cha, K. H. Ko, S. J. Lee, J. W. Han, J. H. Cheon, „An Efficient Implementation of Braid Groups“, Advances in Cryptology - Asiacrypt 2001, LNCS 2248, S. 144-156

[9] J. H. Cheon „Public-key Cryptosystems on Noncommutative Algebraic Structures“, Workshop „Algebraic Methods in Crytography“ des Graduiertesnkollegs der Ruhr-Universität Bochum, 8. und 9. November 2001

[10] J. H. Cheon, B. Jun, „A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem“, Advances in Cryptology - Crypto 2003, LNCS 2729, S. 212-225

[11] B. Chor, R. L. Rivest, „A knapsack-type public key cryptosystem based on arithmetic in finite fields“, Advances in Cryptology - Crypto 1984, LNCS 196, S. 54-65

[12] P. Dehornoy, „Braid-based cryptography“, Preprint, Universität Caen, 2003, Zur Zeit erhältlich unter http://matin.math.unicaen.fr/∼dehornoy/papers.html

[13] W. Diffie, M. E. Hellman, „New directions in cryptography“, IEEE Transactions on Information Theory, Volume 22, 1976, S. 644-654

[14] T. ElGamal, „A public key cryptosystem and a signature scheme based on discrete logarithms“, Advances in Cryptology - Crypto 1984, LNCS 196, S. 10-18

[15] T. ElGamal, „A public key cryptosystem and a signature scheme based on discrete logarithms“, IEEE Transactions on Information Theory, Volume 31, 1985, S. 469-472 Eine frühere Version dieser Arbeit ist [14].

[16] S. Flannery, „Cryptography: An Investigation of a New Algorithm vs. the RSA“, http://cryptome.org/flannery-cp.htm

[17] O. Goldreich, „Foundations of Cryptography - Basic Tools“, Cambridge University Press, 2001

[18] G. Havas, T. Nicholson, „Collection“, 1976 ACM Symposium on Symbolic and Algebraic Computation, SYMSAC '76, S. 9-14

[19] D. Hofheinz, R. Steinwandt, „A Practical Attack on Some Braid Group Based Cryptographic Primitives“, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003, LNCS 2567, Miami, USA, S. 187-198

[20] J. Hughes, „A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem“, 7th Australasien Conference on Information Security and Privacy, ACISP 2002, LNCS 2384, S. 176-189

[21] B. Huppert, „Endliche Gruppen I“, Springer-Verlag, 1967

[22] N. Koblitz, „Elliptic curve cryptosystems“, Mathematics of Computation, Volume 48, 1987, S. 203-209

[23] K.H. Koo, S.J. Lee, J.H. Cheon, J.W. Han, J. Kang, C. Park, „New Public-Key Cryptosystem Using Braid Groups“, Advances in Cryptology - Crypto 2000, LNCS 1880, S. 166-183

[24] D. W. Kravitz, „Digital signature algorithm“, U.S. Patent No. 5,231,668, 27 Jul 1993

[25] H. Kurzweil, „Endliche Gruppen“, Springer Verlag, 1977

[26] R. Laue, J. Neubüser, U. Schoenwaelder, „Algorithms for Finite Soluble Groups and the SOGOS System“, Computational Group Theory, M. Atkinson ed., Academic Press, 1984, S. 105-135

[27] E. Lee, S. J. Lee, S. G. Hahn, „Pseudorandomness from Braid Groups“, Advances in Cryptology - Crypto 2001, LNCS 2139

[28] E. Lee, J. H. Park, „Cryptanalysis of the Public-Key Encryption based on Braid Groups“, Advances in Cryptology - Eurocrypt 2003, LNCS 2656, S. 477-490

[29] S. J. Lee, „The trapdoor onway functions in braid groups“, Workshop „Algebraic Methods in Crytography“ des Graduiertesnkollegs der Ruhr-Universität Bochum, 8. und 9. November 2001

[30] S. J. Lee, E. Lee, „Potential Weaknesses of the Commutator Key Agreement Protocol Based on Braid Groups“, Advances in Cryptology - Eurocrypt 2002, LNCS 2332, S. 14-28

[31] A. K. Lenstra, H. W. Lenstra, „The Development of the Number Field Sieve“, Lecture Notes in Mathematics 1554, Springer-Verlag, 1993

[32] A. K. Lenstra, E. R. Verheul, „Selecting Cryptographic Key Sizes“, Journal of Cryptology, Volume 14(4), 2001, S. 255-293

[33] U. Maurer, „Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms“, Advances in Cryptology - Crypto 1994, LNCS 839, S. 271-281

[34] K. S. McCurley, „The discrete logarithm problem“, Cryptology and Computational Number Theory, Volume 42 of Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1990, S. 49-74

[35] A. Menezes, P. van Oorschot, S. Vanstone, „Handbook of Applied Cryptography“, CRC Press, 1997

[36] R. C. Merkle, M. E. Hellman, „Hiding information and signatures in trapdoor knapsacks“, IEEE Transactions on Information Theory, 24 (1978), S. 525-530

[37] V. S. Miller, „Use of elliptic curves in cryptography“, Advances in Cryptology - Crypto 1985, LNCS 218, S. 417-426

[38] M. Naor, O. Reingold, „Number-Theoretic Constructions of Efficient Pseudo-Random Functions“, Proceedings of the 38th Symposium of Foundations of Computer Science (FOCS) 1997, S. 458-467

[39] A. M. Odlyzko, „Discrete logarithms and smooth polynomials“, Finite Fields: Theory, Applications and Algorithms, Volume 168 of Contemporary Mathematics, American Mathematical Society, 1994, S. 269-278

[40] A. M. Odlyzko, „The rise and fall of knapsack cryptosystems“, Cryptology and Computational Number Theory, Volume 42 of Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1994, S. 75-88

[41] T. Okamoto, D. Pointcheval, „The Gap-Problems: a New Class of Problems for the Security of Cryptographic schemes“, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, LNCS 1992, Cheju Islands, South Korea, S. 104-118

[42] S. H. Paeng, K. C. Ha, J. H. Kim, S. Chee, C. Park, „New Public Key Cryptosystem Using Finite Non Abelian Groups“, Advances in Cryptology - Crypto 2001, LNCS 2139, S.470-485

[43] S. H. Paeng, D. Kwon, K. C. Ha, J. H. Kim „Improved public key cryptosystem using finite non abelian groups“, IACR EPrint-Server, Report 2001/066, http://eprint.iacr.org/2001/066

[44] J. M. Pollard, „Monte Carlo methods for index computation (mod p)“, Mathematics of Computation, Volume 32, 1978, S. 918-924

[45] R. L. Rivest, A. Shamir, L. M. Adleman, „A method for obtaining digital signatures and public-key cryptosystems“, Communications of the ACM, Volume 21, 1978, S. 120-126

[46] C. P. Schnorr, „Efficient identification and signatures for smart cards“, Advances in Cryptology - Crypto 1989, LNCS 435, S. 239-252

[47] C. P. Schnorr, „Efficient signature generation by smart cards“, Journal of Cryptology, Volume 4(4), 1991, S. 161-174
Eine frühere Version dieser Arbeit ist [46].

[48] A. Shamir, „A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem“, Advances in Cryptology - Crypto 1982, Plenum Press, S. 279-288

[49] A. Shamir, „A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem“, IEEE Transactions on Information Theory, 30 (1984), S. 699-704
Eine frühere Version dieser Arbeit ist [48].

[50] D. Shanks, „Class number, a theory of factorization and genera“, Proc. Symp. Pure Math. 20, 1969, AMS, Providence, R.I., 1971, S. 415-440

[51] V. Shoup, „Lower bounds for discrete logarithms and related problems“, Advances in Cryptology - Eurocrypt 1997, LNCS 1233, S. 256-266

[52] H. Sibert, „Algorithmique des groupes de tresses“, Doktorarbeit, Universität Caen, 2003, Zur Zeit erhältlich unter http://www.math.unicaen.fr/∼sibert/indexe.html

[53] C. Tobias „Practical Oblivious Transfer Protocols“, 5th International Workshop on Information Hiding, IH 2002, LNCS 2578, S. 415-426

[54] C. Tobias, „Security Analysis of the MOR Cryptosystem“, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003, LNCS 2567, S. 175-186

[55] C. Tobias, „Security Analysis of MOR using $GL(2, R) \times_\theta \mathbb{Z}_n$“, 2nd International Workshop on Security in Information Systems, WOSIS 2004

[56] C. P. Waldvogel, J. L. Massey, „The probability distribution of the Diffie-Hellman key“, Advances in Cryptology - Auscrypt 1992, LNCS 718, S. 492-504

[57] I. Wegener, „Komplexitätstheorie“, Springer-Verlag, 2003

[58] J. Wolfart, „Einführung in die Zahlentheorie und Algebra“, Vieweg Verlag, 1996

[59] A. Yamamura, „Public key cryptosystems using the modular group“, 1st International Public Key Cryptography Conference PKC 1998, LNCS 1431, S. 203-216

[60] A. Yamamura, „A functional cryptosystem using a group action“, 4th Australian Information Security and Privacy Conference, ACISP 1999, LNCS 1587, S. 314-325

Index