Kryptosystem von Paillier: Analyse und Verbesserungen

Andreas Kumlehn

31. März 2006

Inhalt

1 Einleitung

In dieser Ausarbeitung wird das Paillier-Kryptosystem behandelt. Es handelt sich hierbei um ein Public-Key-Verfahren zur Ver- und Entschlüsselung von Nachrichten. Allgemein bekannte Beispiele für Public-Key-Verfahren sind *RSA* und *ElGamal*. Das vorliegende System wurde in [4] vorgestellt und gilt als ein Kandidat für ein sicheres Verschlüsselungsverfahren.
Diese Ausarbeitung beginnt mit einigen Grundlagen 2, die zum Verständnis der folgenden Ausführungen nötig sind. Um diesen Teil der Ausarbeitung übersichtlich zu gestalten, wird für weitere Grundlagen auf die Ausarbeitung [5] verwiesen.
In Kapitel 3 wird zuerst die Laufzeit und semantische Sicherheit des Paillier-Kryptosystems in seiner ursprünglichen Form untersucht. Kapitel 4 beinhaltet Vorschläge zur Verbesserung der Laufzeit durch Veränderung der Verschlüsselungsfunktion. Dadurch wird eine alternative Entschlüsselungsmöglichkeit eröffnet. Von diesem verbesserten Verfahren (veröffentlicht in [1]) wird anschließend eine komplette Untersuchung vorgenommen.

2 Grundlagen

Einige Notationen werden durchgehend in der gesamten Ausarbeitung genutzt und an dieser Stelle kurz eingeführt.
Die Berechnungen werden in der Ausarbeitung zu einem Großteil in $\mathbb{Z}_{N^2}^*$ ausgeführt, also $\mod n^2$.
Die Zahl n ist dabei das Produkt zweier Primzahlen p, q mit möglichst gleich Bitlänge und wird durchgehend in der Ausarbeitung in dieser Form als gegeben angenommen.
Die Funktion $\lambda(\cdot)$ bezeichnet im Folgenden die Carmichael-Funktion. Für $n = pq$ mit p, q prim gilt $\lambda := \lambda(n) = kgV(p-1, q-1)$ und $\lambda(n^2) = n\lambda(n)$.

2.1 Laufzeiten

Die beiden behandelten Verschlüsselungsverfahren werden hinsichtlich ihrer Laufzeit untersucht. Die genutzten mathematischen Operationen werden alle modular ausgeführt. Die Laufzeiten beziehen sich alle auf die Bitlänge des Moduls (für $\mathbb{Z}_n$ ergibt sich eine Bitlänge von $\log(n)$). Eine Addition hat somit Laufzeit $\mathcal{O}(\log(n))$. Die Laufzeit der modularen Multiplikation ist $\mathcal{O}\left(\log^2(n)\right)$, weil höchstens $\log(n)$ Additionen benötigt werden um zwei Zahlen miteinander zu multiplizieren. Die modulare Exponentiation hat Laufzeit $\mathcal{O}\left(\log^3(n)\right)$, weil sie maximal $\log(n)$ modulare Multiplikationen benötigt. Einige Berechnungen werden in $\mathbb{Z}_{n^2}^*$ ausgeführt. Die Bitlänge dieses Moduls ist somit $\log\left(n^2\right) = 2\log(n)$. Die Laufzeiten der modularen Operationen ändern sich somit in der $\mathcal{O}$-Notation nicht.

2.2 Sicherheit

Um die Sicherheit der Kryptosysteme analysieren zu können werden zunächst einige Begriffe benötigt.

Definition 1. *Eine Funktion $f : \mathbb{N} \mapsto \mathbb{R}^+$ heisst genau dann vernachlässigbar, wenn es für jedes Polynom $P(n)$ ein n_0 gibt mit $f(n) \leq \frac{1}{P(n)}$ für alle $n > n_0$.*

Eine wünschenswerte Eigenschaft eines Kryptosystems ist die semantische Sicherheit.

Definition 2. *Ein Verschlüsselungsverfahren heisst semantisch sicher, wenn ein Angreifer für alle möglichen Verteilungen von Klartexten aus der Verschlüsselung eines zufälligen Klartextes nichts über den Klartext berechnen kann, was er nicht ohne die Verschlüsselung schon berechnen konnte.*

Diese Definition der semantischen Sicherheit ist äquivalent zu dem folgenden Begriff:

Definition 3. *Ein Verschlüsselungsverfahren heißt polynomiell ununterscheidbar, falls für alle Paare von möglichen Nachrichten (m_0, m_1) und einem gegebenen Chiffretext $c = E_g(m_b, r)$ mit $b \in_R \{0, 1\}$ als Verschlüsselung einer der beiden Nachrichten kein effizienter Algorithmus A existiert mit:*

$$Pr(A(c) = b) \geq \frac{1}{2} + \epsilon, \epsilon > \frac{1}{P(n)}$$

Diese beiden Sicherheitsbedingungen sind äquivalent (siehe [3]). Bei den Beweisen zur semantischen Sicherheit in dieser Ausarbeitung wird immer die polynomielle Ununterscheidbarkeit hergeleitet.

3 Analyse

In diesem Kapitel wird das Kryptosystem von Paillier untersucht. Dabei wird auf die Laufzeit von Ver- sowie Entschlüsselung von Nachrichten, allgemeine Parameter und semantische Sicherheit des Systems eingegangen. Die Korrektheit des Systems sowie weitere mathematischen Eigenschaften wurden in [4] bewiesen und werden im Folgenden als gegeben angesehen.
Das Paillier-Kryptosystem ist ein asymmetrisches Verfahren. Zum Setup des Systems müssen zwei Primzahlen berechnet p, q werden. Aus diesen beiden Zahlen können der private Schlüssel (λ) und der öffentliche Schlüssel (n, g) mit $n = pq$ berechnet werden (weitere Erläuterungen siehe 3.1.3). Die Ver- bzw. Entschlüsselung wird jeweils in den folgenden Sektionen erläutert.

3.1 Laufzeitanalyse

3.1.1 Verschlüsselung

Zur Verschlüsselung wird die folgende Funktion genutzt:

$$\begin{aligned} \mathcal{E}_g : \mathbb{Z}_n \times \mathbb{Z}_n^* &\longrightarrow \mathbb{Z}_{n^2}^* \\ (m, r) &\longmapsto g^m r^n \mod n^2 \end{aligned}$$

Wenn g ein Element der Menge $\mathcal{B} = \{w \in \mathbb{Z}_{n^2}^* \mid ord(w) = kn, k = 1, \ldots, \lambda\}$ ist, dann ist $\mathcal{E}_g$ eine Bijektion und damit zur Verschlüsselung geeignet. Das gewählte Element g ist ein Teil des öffentlichen Schlüssel. Das Verschlüsselungsverfahren ergibt sich wie folgt:

ENCRYPTION$((pk(n, g), m \in Z_n))$
1 $r \in_R \mathbb{Z}_n^*$
2 $c \leftarrow \mathcal{E}_g(m, r)$
3 **return** c

Bei der Berechnung von $\mathcal{E}_g(m, r)$ werden folgende Operationen ausgeführt:

	OPERATION		LAUFZEIT
1	Exponentiation	$g^m \mod n^2$	$\mathcal{O}\left(\log^2(n) \log(m)\right)$
2	Exponentiation	$r^n \mod n^2$	$\mathcal{O}\left(\log^3(n)\right)$
3	Multiplikation	$r^n g^m \mod n^2$	$\mathcal{O}\left(\log^2(n)\right)$

Die Exponentiation in Zeile 2 dominiert somit die Laufzeit der Verschlüsselung und es ergibt sicht eine Gesamtlaufzeit von $\mathcal{O}\left(\log^3(n)\right)$.

3.1.2 Entschlüsselung

Zur Entschlüsselung eines Chiffretextes wird eine weitere Funktion benötigt:

$$\begin{aligned} L : \mathcal{S}_n &\longrightarrow \mathbb{Z}_n^* \\ (u) &\longmapsto \frac{u-1}{n} \end{aligned}$$

mit $\mathcal{S}_n = \{u < n^2 | u = 1 \mod n\}$.
Des weiteren gilt nach Satz von Carmichael für ein beliebiges $w \in \mathbb{Z}_{n^2}^* : w^{n\lambda} = 1 \mod n^2$ und somit $w^\lambda \in \mathcal{S}_n$ (siehe [5]).
Nun kann die Entschlüsselung formuliert werden:

DECRYPTION$(sk(\lambda), c \in \mathbb{Z}_{n^2}^*)$
1 $u_1 \leftarrow c^\lambda \mod n^2$
2 $x \leftarrow L(u_1)$
3 $y \leftarrow L(g^\lambda \mod n^2)^{-1} \mod n$
4 $m \leftarrow xy \mod n$
5 **return** m

Die Laufzeit der Funktion $L(u)$ beläuft sich auf $\mathcal{O}\left((\lceil \log(n) \rceil)^2\right)$, weil der Wert $n^{-1} \mod 2^{\lceil \log(n) \rceil}$ vorab berechnet werden kann. Es muss also nur eine Multiplikation $\mod 2^{\log(n)}$ berechnet werden. Ebenso kann der Wert $y = L(g^\lambda \mod n^2) \mod n$ aus Schritt 3 vorberechnet werden, da g und n durch Wahl des öffentlichen Schlüssels festgelegt werden. Es verbleiben die folgenden Operationen:

	OPERATION	LAUFZEIT	
1	Exponentiation	$c^\lambda \mod n^2$	$\mathcal{O}\left(\log^2(n)\log(\lambda)\right)$
2	Multiplikation	$L(u_1)$	$\mathcal{O}\left((\lceil \log(n) \rceil)^2\right)$
4	Multiplikation	$m = xy \mod n$	$\mathcal{O}\left(\log^2(n)\right)$

Bei der Entschlüsselung wird die Laufzeit durch die Berechnung der Exponentiation in Schritt 1 dominiert. Es folgt eine Laufzeit von $\mathcal{O}\left(\log^2(n)\log(\lambda)\right)$.

3.1.3 allgemeine Parameter

Zu den allgemeinen Parametern eines Kryptosystems gehören die Nachrichtenexpansion und der Aufwand für das Setup.
Die Nachrichtenexpansion gibt an, um welchen Faktor ein Chiffretext länger ist als ein Klartext. In unserem Fall sind die Nachrichten aus $\mathbb{Z}_n$ und haben somit eine Bitlänge von $\log(n)$. Die Elemente des Chriffretextraums $\mathbb{Z}_{n^2}^*$ haben eine Bitlänge von $2\log(n)$. Es folgt somit ein Faktor „2“ für die Nachrichtenexpansion.
Wie am Anfang dieses Kapitels bereits erwähnt, müssen für das Setup des Systems zwei Primzahlen p, q bestimmt werden. Dieses ist analog zum Setup von RSA. Die Erzeugung der Primzahlen kann z.B. mit dem probabilistischen Primzahltest von Miller-Rabin erfolgen (siehe [2], Seiten 890 - 896).
Die Wahl des Elementes $g \in \mathcal{B}$ kann in konstanter Zeit erfolgen, weil einige Elemente aus $\mathcal{B}$ bekannt sind (z.B. Einheitswurzeln $\mod n^2$).

3.2 semantische Sicherheit

Die semantische Sicherheit des Paillier-Kryptosystem basiert auf dem folgenden Entscheidungsproblem:

Definition 4. *Gegeben eine Zahl $x \in \mathbb{Z}^*_{n^2}$ ist zu entscheiden, ob x ein n-ter Rest ist oder nicht. Dieses Problem wird im folgenden mit $CR[n]$ für „Composite Residuosity“ bezeichnet.*

Dieses ist ein gut untersuchtes Problem und derzeit ist kein effizienter Algorithmus zur Lösung des Problems bekannt. Die semantische Sicherheit des Verschlüsselungssystems beruht damit auf folgender Annahme:

Annahme 1. *Das Problem $CR[n]$ ist nicht in polynomialzeit lösbar. Das heisst: es gibt keinen polynomialzeit Unterscheider um n-te Reste $\mod n^2$ von nicht-n-ten Resten $\mod n^2$ zu unterscheiden.*

Diese Annahme wird in der Literatur mit $DCRA$ für „Decisional Composite Residuosity Assumption“ abgekürzt. Basierend auf der vorherigen Annahme kann nun die semantische Sicherheit des Systems bewiesen werden:

Theorem 1. *Das Paillier-Kryptosystem ist genau dann semantisch sicher, wenn die „Decisional Composite Residuosity Assumption“ hält.*

Beweis. Seien zwei Nachrichten $m_0, m_1 \in \mathbb{Z}_n, m_0 \neq m_1$ und die Verschlüsselung $c \in \mathbb{Z}^*_{n^2}$ einer der beiden Nachrichten gegeben. Aus dem öffentlichen Schlüssel kennen wir das Element $g \in \mathbb{Z}^*_{n^2}$, welches bei der Verschlüsselung genutzt wurde. Betrachten wir nun cg^{-m_0} in Abhängigkeit der beiden möglichen Klartexte:

$$m_0 : cg^{-m_0} = \mathcal{E}_g(m_0, r)g^{-m_0} = g^{m_0}g^{-m_0}r^n \mod n^2 = r^n \mod n^2$$
$$m_1 : cg^{-m_0} = \mathcal{E}_g(m_1, r)g^{-m_0} = g^{m_1}g^{-m_0}r^n \mod n^2 = g^{m_1-m_0}r^n \mod n^2$$

Aus dem ersten Fall (c ist Verschlüsselung von m_0) folgt, dass cg^{-m_0} ein n-ter Rest $\mod n^2$ ist. Im zweiten Fall cg^{-m_0} gerade kein n-ter Rest, weil $0 \neq m_1 - m_0 \neq n$ gilt (Folgerung aus Klartextraum $\mathbb{Z}_n$).
Die beiden Richtungen des Beweises ergeben sich somit wie folgt:

- $\Rightarrow$ Wenn das Paillier-Kryptosystem semantisch sicher ist, dann gibt es bei bekannten Nachrichten m_0, m_1 und einer Verschlüsselung c keinen effizienten Algorithmus zur Unterscheidung, ob c die Verschlüsselung von m_0 oder von m_1 ist. Wenn es einen solchen Algorithmus nicht gibt, kann es nach obigen Betrachtungen auch keinen Algorithmus geben, welcher n-te Reste von nicht-n-ten Resten effizient unterscheidet. Somit folgt die Annahme $DCRA$.

- $\Leftarrow$ Wenn die Annahme $DCRA$ gilt, dann kann nicht effizient zwischen n-ten Resten und nicht-n-ten Resten unterschieden werden. Es kann somit auch nicht bestimmt werden, ob cg^{-m_0} ein n-ter Rest ist oder nicht. Aus der gegebenen Verschlüsselung c kann also nicht effizient bestimmt werden, ob m_0 oder m_1 verschlüsselt wurde und das System ist semantisch sicher.

4 Verbesserungen

Dieses Kapitel umfasst einige Optimierungen am Kryptosystem von Paillier. Dies geschieht durch die Wahl eines speziellen Elementes g für die Verschlüsselungsfunktion $\mathcal{E}_g$, welches weitere Veränderungen ermöglicht. Die neue Verschlüsselungsfunktion wird auf ihre kryptografische Brauchbarkeit untersucht. Außerdem wird ein alternatives Entschlüsselungsverfahren ermöglicht. Insgesamt ergeben sich Verbesserungen in der Laufzeit des Systems und abschließend wird die semantische Sicherheit dieses optimierten Verfahrens erläutert.
Die Optimierungen wurden in [1] vorgestellt.

4.1 Verschlüsselung

4.1.1 Veränderungen

Zur Verschlüsselung wird erneut die bereits bekannt Funktion $\mathcal{E}_g$ genutzt. Das Element g stammt dabei aus der Menge $\mathcal{B} = \{w \in \mathbb{Z}^*_{n^2} \mid ord(w) = kn, k = 1, \ldots, \lambda\}$. In der Veröffentlichung [4] wurde gezeigt, dass die Komplexität eines Angriffes auf die Funktion $\mathcal{E}_g$ unabhängig vom gewählten Element $g \in \mathcal{B}$ ist. Es bleibt nun ein spezielles Element zu suchen, welches Verbesserungen ermöglicht.
Die n-ten Einheitswurzeln $\mod n^2$ (siehe [5]) sind Elemente der Menge $\mathcal{B}$, weil sie die Ordnung n haben. Wir wählen nun den Generator $g = (1 + n)$ der n-ten Einheitswurzeln $\mod n^2$ aus und setzen dieses g in die Verschlüsselung ein:

$$\begin{aligned}\mathcal{E}_g(m, r) = \mathcal{E}_{(1+n)}(m, r) &= (1 + n)^m r^n \mod n^2 \\ &= (1 + mn)r^n \mod n^2 \\ &= r^n + mnr^n \mod n^2\end{aligned}$$

Diese Darstellung der verschlüsselten Nachricht ermöglicht einen alternativen Entschlüsselungsvorgang, welcher in 4.2 beschrieben wird. Diese Alternative ermöglicht es, den Exponenten n am zufälligen Element $r \in \mathbb{Z}^*_n$ zu verändern. Der Exponent n wird in dem ursprünglichen System benötigt, um bei der Entschlüsselung einer Nachricht den Satz von Carmichael anwenden zu können $((r^n)^\lambda = (r^{n\lambda} = 1 \mod n^2)$.
Wir wählen nun einen neuen Exponenten $e \in \mathbb{Z}_n$ mit $\gcd(e, \lambda(n^2)) = 1$ [1]. Dieser Exponenten e sollte deutlich kleiner als n gewählt werden ($e << n$), um einen möglichst große Verbesserung bei der Laufzeit der Verschlüsselung zu erhalten.
Der Index der Verschlüsselungsfunktion beinhaltet im Folgenden den gewählten Exponenten e, weil das Element g fest auf den Wert $(1 + n)$ gesetzt wurde. Insgesamt ergibt sich somit eine neue Verschlüsselungsfunktion:

$$\begin{aligned}\mathcal{E}'_e : \mathbb{Z}_n \times \mathbb{Z}^*_n &\longrightarrow \mathbb{Z}^*_{n^2} \\ (m, r) &\longmapsto r^e + mnr^e \mod n^2\end{aligned}$$

[1] Erläuterungen siehe Entschlüsselung 4.2

Der Verschlüsselungsalgorithmus ist äquivalent zu dem des nicht optimierten Systems (siehe 3.1.2), nur die Funktion $\mathcal{E}_g(m,r)$ wird durch $\mathcal{E}_e^{'}(m,r)$ ersetzt. Als öffentlicher Schlüssel ergibt sich (n,e) mit $n = pq$.

4.1.2 Eigenschaften

In diesem Teil werden die kryptografischen Eigenschaften der veränderten Verschlüsselung untersucht. Hierzu wird gezeigt, dass die Funktion $\mathcal{E}_e^{'}(m,r)$ eine Permutation ist. Des weiteren wird mit Hilfe einer Annahme gezeigt, dass es sich außerdem um eine Einwegfunktion handelt, die ohne die Kenntnis eines Geheimnisses schwer zu invertieren ist. Die Funktion $\mathcal{E}_e^{'}(m,r)$ ist offensichtlich effizient auszuwerten bei gegebenen Eingaben m, r und zusammen mit den vorherigen Betrachtungen folgt dann letztendlich, dass die optimierte Verschlüsselung für kryptografische Anwendungen geeignet ist. Betrachten wir nun zuerst die Abbildungseigenschaften der Funktion:

Lemma 1. *Die Funktion $\mathcal{E}_e^{'}$ ist eine Permutation.*

Beweis. Betrachten wir zuerst die Signatur der Funktion: $\mathcal{E}_e^{'} : \mathbb{Z}_n \times \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_{n^2}^*$. Es gilt hierbei, dass der Werte- und der Bildbereich der Funktion die gleiche Anzahl von Elementen haben ($|\mathbb{Z}_n \times \mathbb{Z}_n^*| = |\mathbb{Z}_n|\,|\mathbb{Z}_n^*| = n\Phi(n) = \left|\mathbb{Z}_{n^2}^*\right|$). Es reicht also zu zeigen, dass $\mathcal{E}_e^{'}$ injektiv ist.
Wir nehmen dafür an, dass es zu einem Bild (Chiffretext) zwei verschiedene Urbilder (Klartexte und zufällige Elemente) gibt:

$$r^e + mnr^e = a^e + bna^e \mod n^2.$$

Nun betrachten wir dieses Bild $\mod n$:

$$r^e = a^e \mod n.$$

Es gilt somit $r = a$, weil beide Elemente aus $\mathbb{Z}_n^*$ ausgewählt wurden. Somit folgt insgesamt auch $m = b$ und erhalten einen Widerspruch zu der vorherigen Annahme. Die Funktion $\mathcal{E}_e^{'}(m,r)$ ist somit injektiv und folglich auch eine Permutation.

Als nächstes betrachten wir die Invertierung der Funktion $\mathcal{E}_e^{'}(m,r)$. Diese sollte ohne der Kenntnis eines Geheimnisses schwer sein, damit das Kryptosystems sicher ist. Der Beweis dieser Eigenschaft beruht erneut auf der Annahme, dass ein gut untersuchtes mathematisches Problem, für welches bis heute kein effizienter Algorithmus existiert, schwer lösbar ist. In diesem Fall handelt sich um das folgende Problem:

Definition 5. *Gegeben $y = x^e \mod n^2, y \in \mathbb{Z}_{n^2}^*$. Berechne $x \in \mathbb{Z}_n$.*

Dieses Problem heißt „Berechnung kleiner e-ter Wurzeln $\mod n^2$“. Zu beachten ist, dass für die Wurzel $x \in \mathbb{Z}_n$ gilt, nicht $x \in \mathbb{Z}_{n^2}^*$.
Dieses Problem entspricht einer Variante der allgemein bekannten RSA-Verschlüsselung $\mod n^2$.
Auf dem Problem beruht die Annahme:

Annahme 2. *Die Berechnung kleiner e-ter Wurzeln $\mod n^2$ ist schwer, d.h. für jeden polynomialzeit beschränkten Algorithmus A gibt es eine vernachlässigbare Funktion f mit:*

$$Pr(A(y,n) = x) = f(n) \text{ bei Eingabe } y = x^e \mod n^2.$$

Mit Hilfe der Annahme über die Berechnung kleiner e-ter Wurzeln $\mod n^2$ wird nun die gewünschte Einwegeigenschaften der Verschlüsselungsfunktion gezeigt.

Lemma 2. *Wenn die Annahme 2 hält, dann ist die Funktion $\mathcal{E}_e'$ eine Einwegfunktion. Das zur Invertierung benötigte Geheimnis ist die Faktorisierung von n.*

Beweis. Der Beweis erfolgt per Reduktion der Berechnung kleiner e-ter Wurzeln auf die Umkehrung der Funktion $\mathcal{E}_e'(m,r)$.
Annahme: Es gibt einen Algorithmus INV, der bei Eingabe einer korrekten Verschlüsselung $(1+mn)r^e$ das Urbild (m,r) effizient berechnen kann mit $Pr(INV((1+mn)r^e)) = (m,r) > f(n)$ mit f vernachlässigbar.
Wir konstruieren jetzt einen Algorithmus A, der den Algorithmus INV ausnutzt, um das Problem der Berechnung kleiner e-ter Wurzeln zu lösen:

A($y \in \mathbb{Z}_{n^2}^*$ mit $y = x^e \mod n^2$)
1 $m \in_R \mathbb{Z}$
2 $c \leftarrow y(1+mn) \mod n^2$
3 $(m,x) \leftarrow \text{INV}(c)$
4 **return** x

Für die Erfolgswahrscheinlichkeit des Algorithmus A folgt dann:

$$Pr(A(y) = x) = Pr(INV(y)) > f(n).$$

Dies ist ein Widerspruch zu der Annahme über die Berechnung kleiner e-ter Wurzeln und somit ist die Funktion $\mathcal{E}_e'$ eine Einwegfunktion. Die Falltür zur effizienten Umkehrung ist die Faktoriserung von n (siehe 4.2).

Die angepasste Funktion $\mathcal{E}_e'(m,r)$ weist somit alle benötigten Eigenschaften für eine kryptografische Verschlüsselungsfunktion auf.

4.2 Entschlüsselung

Durch die Veränderungen der Verschlüsselungsfunktion wird ein alternativer Weg der Entschlüsselung eröffnet. Dieser ermöglicht zusammen mit den vorherigen Anpassungen eine geringere Laufzeit des verbesserten Kryptosystems.
Betrachten wir zur Herleitung der neuen Entschlüsselung nun zuerst einen Chriffretext $\mod n$:

$$\begin{aligned}\mathcal{E}_e(m,r) &= r^e + mnr^e \mod n^2 \\ &\equiv r^e \mod n\end{aligned}$$

Aus dem Term $r^e \mod n$ lässt sich mit Hilfe eines Exponenten d mit $ed = 1 \mod \Phi(n)$ das Zufallselement r bestimmen. Dieses Berechnung erspricht der Entschlüsselung der RSA-Funktion und daraus leitet sich auch der geheime Schlüssel des verbesserten Systems ab: (e,d,n) mit $ed = 1 \mod \Phi(n)$. Der Exponent e erfüllt nach Wahl die Eigenschaft $\gcd(e, \lambda(n^2)) = 1$ und ist deswegen insbesondere $\mod \Phi(n)$ invertierbar. Die Zahl d kann deswegen nach Wahl von e während des Setups des Kryptosystems berechnet werden. Dazu wird die Faktorisierung von $n = pq$ benötigt, welche somit auch die Falltür für das Brechen des beschleunigten Kryptosystems darstellt.
Der gesamte Entschlüsselungsalgorithmus arbeitet anschließend wie folgt:

OptDecryption($sk(e,d,n), c \in \mathbb{Z}_{n^2}^*$)
1 $r = c^d \mod n$
2 $r^{-e} \leftarrow EEA(r^e \mod n^2)$
3 $m \leftarrow L(cr^{-e}) \mod n$
4 **return** m

EEA meint hierbei die Anwendung des erweiterten euklidischen Algorithmus zur Berechnung des multiplikativen Inversen von $r^e \mod n^2$.
Die Korrektheit der Entschlüsselung ergibt sich bei Eingabe $c = (1+mn)r^e \mod n^2, m \in \mathbb{Z}_n, r \in \mathbb{Z}_n^*$ aus der Konstruktion. Im ersten Schritt der Entschlüsselung wird der korrekte Wert des zufälligen Elementes r berechnet, weil $r \in \mathbb{Z}_n^*$ gilt. Anschließend wird $r^e \mod n^2$ berechnet und invertiert ($\gcd(r,n) = 1 \Rightarrow \gcd(r,n^2) = 1$). Nach der Multiplikation cr^{-e} verbleibt $(1+mn)$ als Eingabe für die Funktion L (siehe 3.1.2). Die Eingabe hat hierbei die benötigte Form. Als Ausgabe der Funktion L ergibt sich der gesuchte Klartext m.

4.3 Laufzeitanalyse

In diesem Teil wird die Laufzeit des verbesserten Kryptosystems analysiert und abschließend mit der Laufzeit des originalen Paillier-Kryptosystems verglichen.

4.3.1 Verschlüsselung

Betrachten wir zuerst die Verschlüsselung anhand der bei der Berechnung von $\mathcal{E}_e'(m,r)$ genutzten Operationen:

	Operation		Laufzeit
1	Exponentiation	$r^e \mod n^2$	$\mathcal{O}\left(\log^2(n)\log(e)\right)$
2	Multiplikation	$mn \mod n^2$	$\mathcal{O}\left(\log^2(n)\right)$
3	Addition	$(1+mn) \mod n^2$	$\mathcal{O}\left(\log(n)\right)$
4	Multiplikation	$(1+mn)r^e \mod n^2$	$\mathcal{O}\left(\log^2(n)\right)$

Die Laufzeit der optimierten Verschlüsselung wird durch die Exponentiation in Zeile 1 dominiert und es ergibt sich die Gesamtlaufzeit $\mathcal{O}\left(\log^2(n)\log(e)\right)$.

4.3.2 Entschlüsselung

Die Laufzeitanalyse der Entschlüsselung ergibt sich wie folgt:

	Operation		Laufzeit
1	Exponentiation	$c^d \mod n$	$\mathcal{O}\left(\log^2(n)\log(d)\right)$
2	Exponentiation	$r^e \mod n^2$	$\mathcal{O}\left(\log^2(n)\log(e)\right)$
3	Invertierung EEA	$r^{-e} \mod n^2$	$\mathcal{O}\left(\log^2(n)\right)$
4	Multiplikation	$cr^{-e} \mod n^2$	$\mathcal{O}\left(\log^2(n)\right)$
5	Multiplikation	$L(cr^{-e})$	$\mathcal{O}\left((\lceil\log(n)\rceil)^2\right)$

Bei der Entschlüsselung ergibt sich die Gesamtlaufzeit $\mathcal{O}\left(\log^2(n)\log(d)\right)$ aus der ersten Zeile. Diese Zeile dominiert die Exponentiation in Zeile 2 unter der Annahme, dass der Exponent e möglichst klein gewählt wurde um eine schnelle Verschlüsselung zu gewährleisten.

4.3.3 Vergleich der beiden Systeme

Betrachten wir nun die Laufzeiten von Ver- bzw. Entschlüsselung der beiden Systeme im Vergleich:

	Verschlüsselung	Entschlüsselung
Originalsystem	$\mathcal{O}\left(\log^3(n)\right)$	$\mathcal{O}\left(\log^2(n)\log(\lambda)\right)$
Optimierung	$\mathcal{O}\left(\log^2(n)\log(e)\right)$	$\mathcal{O}\left(\log^2(n)\log(d)\right)$

Die Laufzeit der Verschlüsselung ist somit annähernd quadratisch in $\log^2(n)$ für kleine Exponenten e. Die Veränderung bei der Entschlüsselung hängt von der Wahl von e und somit d ab.

4.4 semantische Sicherheit

Abschließend betrachten wir nun die semantische Sicherheit des beschleunigten Kryptosystems. Hierzu benötigen wir erneut eine Annahme über ein mathematisches Problem, damit wir die Sicherheit beweisen können. Es handelt sich hierbei um eine Entscheidungsvariante des Problems der Berechnung kleiner e-ter Wurzeln (siehe 2), welches wir zuvor für den Beweis der Einwegeigenschaften der Verschlüsselung formuliert haben.

Definition 6. *Gegeben ein Elemente $x \in \mathbb{Z}^*_{n^2}$, entscheide ob x ein e-ter Rest $\mod n^2$ ist oder nicht.*

Dieses Problem wird mit „Entscheidung kleiner e-ter Reste“ bezeichnet. Für die Formalisierung der Annahme definieren wir für einen beliebigen probabilistischen polynomialzeit beschränkten Algorithmus A die folgenden Verteilungen:

1. $P_{random} = Pr(A(x, n) = 1)$ mit Eingabe $x \in_R \mathbb{Z}^*_{n^2}$
2. $P_{residue} = Pr(A(y, n) = 1)$ mit Eingabe $y = x^e \mod n^2, x \in_R \mathbb{Z}_n$

mit zufällig gewählten Zahlen $n = pq$ und Exponent e mit $\gcd(e, \lambda(n^2))$.
Die Wahrscheinlichkeitsverteilung P_{random} behandelt den Fall, dass der Algorithmus A für ein beliebiges Element $x \in_R \mathbb{Z}^*_{n^2}$ ausgibt, dass x ein e-ter Rest $\mod n^2$ ist. In der Verteilung $P_{residue}$ wird die korrekte Erkennung eines e-ten Restes $\mod n^2$ behandelt.

Annahme 3. *Die Verteilungen P_{random} und $P_{residue}$ sind ununterscheidbar, d.h. für jeden probabilistischen polynomialzeit beschränkten Algorithmen A existiert eine vernachlässigbare Funktion $f(\log(n))$ mit:*

$$|P_{random} - P_{residue}| < f(log(n)).$$

Diese Annahme wird im Folgenden mit DSERA für „Decisional Small e-th Residues Assumption“ bezeichnet.

Um die semantische Sicherheit des Kryptosystems zu zeigen, erfolgt zuerst der Beweis zweier Lemmata aus denen die gewünschte Aussage direkt folgt.

Lemma 3. *Wenn die Annahme DSERA nicht hält, dann ist die Verschlüsselung $\mathcal{E}'_e(m, r)$ nicht semantisch sicher.*

Beweis. Nehmen wir nun an, dass die Annahme $DSERA$ nicht hält, d.h. es gibt einen Algorithmus A mit $|P_{random} - P_{residue}| \geq f(log(n))$ und f nicht vernachlässigbar. Um die semantische Sicherheit der Verschlüsselung zu widerlegen müssen wir ein Paar von Nachrichten m_0, m_1 konstruieren, so dass ein probabilistischer Algorithmus B aus der Verschlüsselung $c = \mathcal{E}'_e(m_b, r)$ mit $b \in_R \{0, 1\}$ das Bit b mit einem nicht vernachlässigbarem Vorteil gegenüber gleichverteiltem Raten bestimmen kann.
Eine dieser beiden Nachrichten legen wir fest: $m_0 = 0$. Wenn die Nachricht m_0 verschlüsselt wird, dann haben die möglichen Chriffretexte die folgende Form:

$$\mathcal{E}'_e(m_0, r) = \mathcal{E}'_e(0, r) = r_0^e \mod n^2 \text{ mit } r_0 \in \mathbb{Z}^*_n.$$

Diese Verschlüsselungen sind somit alle e-te Reste $\mod n^2$ und der Algorithmus A erkennt diese Elemente mit Wahrscheinlichkeit analog zu $P_{residue}$.

Die zweite Nachrichte m_1 müssen wir aus der Menge der verbleibenden möglichen Nachrichten $\mathbb{Z}_n \backslash \{0\}$ auswählen. Die möglichen Verschlüsselungen all dieser Nachrichten $m^{'}$ haben die folgende Form:

$$c^{'} = \mathcal{E}_e^{'}(m^{'}, r) = r_1^e + r_1^e n m^{'} \mod n^2 \text{ mit } r_1 \in \mathbb{Z}_n^*.$$

All diese möglichen Verschlüsselungen $c^{'}$ der verbleibenden Nachrichten sind keine e-te Reste $\mod n^2$. Wäre $c^{'}$ ein e-ter Rest, dann gilt: $\exists a \in \mathbb{Z}_n^* : a^e \mod n^2 = c^{'} = r_1^e + r_1^e n m^{'} \mod n^2$. Dies ist ein Widerspruch zur Permutationseigenschaft, insbesondere der Injektivität von $\mathcal{E}_e^{'}(m, r)$.
Aus der Annahme des Beweises, dass die Annahme $DSERA$ nicht hält, ergibt sich, dass der Algorithmus A die beiden Verteilungen P_{random} und $P_{residue}$ effizient unterscheiden kann mit nicht vernachlässigbarer Wahrscheinlichkeit. Hieraus folgt, dass es eine Nachricht $m_1 \in \mathbb{Z}_n \backslash \{0\}$ geben muss, für welche die Unterscheidung effizient möglich ist.
Für diese beiden Nachrichten m_0 und m_1 kann der Algorithmus A somit bei gegebener Verschlüsselung $c = \mathcal{E}_e^{'}(m_b, r)$ mit $b \in_R \{0, 1\}$ das Bit b mit nicht vernachlässigbarer Wahrscheinlichkeit $f(\log(n))$ bestimmen. Damit ist die Verschlüsselung $\mathcal{E}_e^{'}(m, r)$ nicht semantisch sicher.

Lemma 4. *Wenn die Verschlüsselung $\mathcal{E}_e^{'}(m, r)$ nicht semantisch sicher, dann gilt die Annahme $DSERA$ nicht.*

Beweis. Angenommen, die Verschlüsselung $\mathcal{E}_e^{'}(m, r)$ ist nicht semantisch sicher. Es existiert somit ein polynomialzeit Algorithmus A, der sich zwei beliebige Nachrichten $m_0, m_1, m_0 \neq m_1$ aussuchen darf. Anschließend übergibt er diese beiden Nachrichten an ein Orakel. Dieses Orakel verschlüsselt eine der beiden Nachrichten und wählt dabei zufällig gleichverteilt aus. Der Algorithmus A erhält also vom Orakel einen Chriffretext $c = \mathcal{E}_e^{'}(m_b, r), b \in_R \{0, 1\}$. Anschließend kann der Angreifer A das Bit b mit folgender Erfolgswahrscheinlichkeit aus c, m_0, m_1 mit $c = \mathcal{E}_e^{'}(m_b, r), b \in_R \{0, 1\}$ bestimmen:

$$Pr(A(c, m_0, m_1) = b) = Pr(A(\mathcal{E}_e^{'}(m_b, r), m_0, m_1) = b)) > \frac{1}{2} + f(\log(n))$$

Die Funktion f sei hierbei nicht vernachlässigbar. Dieser Angreifer A hat somit einen echten Vorteil beim Bestimmen des Bit b aus dem Chriffretext c gegenüber dem zufällig gleichverteilten Raten des Bits (dieses hätte Erfolgswahrscheinlichkeit $\frac{1}{2}$).
Mit Hilfe dieses Angreifers A können wir nun den folgenden Algorithmus D konstruieren, um für ein gegebenes Element $z \in \mathbb{Z}_{n^2}^*$ zu entscheiden, ob z ein e-ter Rest ist oder nicht. Dieser konstruierte Algorithmus D ersetzt das vom Angreifer A genutzte Orakel und erhält als Eingabe die zu untersuchende Zahl z sowie die beiden von A gewählten Nachrichten m_0, m_1.

D(z, m_0, m_1)
1 $b \in_R \{0, 1\}$
2 $c \leftarrow z(1 + m_b n) \mod n^2$
3 $\bar{b} = A(c, m_0, m_1)$
4 **if** $(b = \bar{b})$
5 **then return** „Yes“
6 **else return** „No“

Der Algorithmus D konstruiert hierbei die Verschlüsselung c aus der Eingabe. Falls z ein e-ter Rest mod n^2 ist, dann gibt es ein $x \in \mathbb{Z}_n$ mit $z = x^e \mod n^2$. Es gilt also für den konstruierten Chriffretext $c = z(1 + m_b n) \mod n^2 = x^e(1 + m_b n) \mod n^2$. Dies ist insbesondere ein gültiger Chriffretext und der Algorithmus A erkennt das gewählte Bit b mit Wahrscheinlichkeit $P_1 > \frac{1}{2} + f(\log(n))$ mit f nicht vernachlässigbar. Die Überprüfung $b = \bar{b}$ trifft somit mit der Wahrscheinlichkeit P_1 zu und

insgesamt hat der Algorithmus D somit Erfolgswahrscheinlichkeit P_1 zur korrekten Erkennung eines e-ten Restes.
Betrachten wir nun den Fall, dass die Zahl z kein e-ter Rest $\mod n^2$ ist. Zu dem Wert $z \in \mathbb{Z}_{n^2}^*$ gibt es Werte $r^{'} \in \mathbb{Z}_n^*, s^{'} \in \mathbb{Z}_n$ mit $z = \mathcal{E}_e^{'}(s^{'}, r^{'})$, weil $\mathcal{E}_e^{'}$ eine Permutation ist und somit insbesondere surjektiv. Es folgt für den von D konstruierten Chiffretext c:

$$\begin{aligned}
c &= z(1 + m_b n) \mod n^2 \\
&= \mathcal{E}_e^{'}(s^{'}, r^{'})(1 + m_b n) \mod n^2 \\
&= (r^{'})^e((1 + s^{'} n)(1 + m_b n) \mod n^2 \\
&= (r^{'})^e(1 + s^{'} n + m_b n + s^{'} m_b n^2) \mod n^2 \\
&= (r^{'})^e(1 + n(s^{'} + m_b + s^{'} m_b n)) \mod n^2 \\
&= (r^{'})^e(1 + n(s^{'} + m_b \mod n)) \mod n^2
\end{aligned}$$

Durch die Addition $s^{'} + m_b$ kann der Angreifer A das Bit b nur durch gleichverteiltes Raten bestimmen und hat somit eine Erfolgswahrscheinlichkeit $P_2 = \frac{1}{2}$.
Betrachten wir nun die Erfolgswahrscheinlichkeit des konstruierten Algorithmus D für beliebige Eingaben $z \in \mathbb{Z}_{n^2}^*$. z kann somit ein e-ter Rest oder ein e-ter Nichtrest sein. Für den Erfolg von D gilt dann:

$$\begin{aligned}
&P(D \text{ erkennt Eingabe } z \in_R \mathbb{Z}_{n^2}^* \text{ korrekt}) \\
&= (\frac{\sharp e\text{-te Reste} \mod n^2}{\Phi(n)n} \cdot P_1) + (\frac{\sharp \text{nicht } e\text{-te Reste} \mod n^2}{\Phi(n)n} \cdot P_2) \\
&= \frac{n}{\Phi(n)n} P_1 + \frac{\Phi(n)n - n}{\Phi(n)n} P_2 \\
&= \frac{1}{\Phi(n)} P_1 + (1 - \frac{1}{\Phi(n)}) P_2 \\
&= \frac{1}{\Phi(n)} P_1 + P_2 - \frac{1}{\Phi(n)} P_2
\end{aligned}$$

Die Terme mit Faktor $\frac{1}{\Phi(n)}$ sind vernachlässigbar in $\log(n)$ und somit bleibt eine Wahrscheinlichkeit von $P_2 = \frac{1}{2}$ erhalten. Der Nenner $\Phi(n)n$ ergibt sich aus der Anzahl Elemente in $\mathbb{Z}_{n^2}^*$.
Insgesamt ergibt sich für den probabilistischen polynomialzeit beschränken Algorithmus D:

$$|P_{random} - P_{residue}| \geq f(log(n))$$

mit f nicht vernachlässigbar nach Annahme des Beweises. Somit hält die Annahme $DSERA$ nicht.

Korollar 1. *Die Verschlüsselung $\mathcal{E}_e^{'}(m, r)$ ist genau dann semantisch sicher, wenn die Annahme $DSERA$ hält.*

Die beiden vorherigen Lemmata behandeln jeweils Hin- bzw. Rückrichtung dieses Korollars.

Literaturverzeichnis

[1] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's cryptosystem revisited. *Proceedings of the 8th ACM Conference on Computer and Communications Security, Pages 206*, 2001.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, 2003.

[3] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences, Volume 28, Issue 2, Pages 270-299*, 1984.

[4] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science, Volume 1592, Page 223*, 1999.

[5] T. Volkhausen. Paillier cryptosystem: A mathematical introduction. *Seminar Public-Key Kryptographie (WS 05/06) bei Prof. Dr. J. Blömer*, 2006.