Distributed secret sharing and threshold decryption with the RSA-based Paillier crypto system without trusted dealer

Silvan Weber
silvan.weber@msc.htwchur.ch

Master of Science in Engineering
Software and Systems
Information and Communication Technologies
E-Voting Seminar
Hochschule für Technik und Wirtschaft HTW Chur, Schweiz
www.htwchur.ch/mse

22. Juni 2011

Abstract. In diesem Paper wird das auf RSA basierende Paillier-Kryptosystem für secret sharing und threshold decryption ohne eine zentrale Stelle untersucht. Eine zentrale Stelle hat gewisse Nachteile, aber das Weglassen einer solchen schafft neue Probleme, was in diesem Bericht ebenfalls diskutiert wird. Es wird auf mehrere Paper, die bereits Vorarbeiten in dieser Richtung gemacht haben, Bezug genommen. Zuletzt wird eine Methode inklusive eines Rechenbeispiel präsentiert, welche ein solches System gänzlich implementiert.

Schlüsselwörter: E-Voting, Paillier, secret sharing, threshold decryption, distributed, without trusted Dealer, BGW, Pedersen

1 Einleitung

Das Ziel dieses Papers ist die Untersuchung des Paillier-Kryptosystems mit der Anwendung auf E-Voting. Im Speziellen wird behandelt, wie ein E-Voting System, das auf dem Paillier-Kryptosystem basiert, ohne zentrale, vertrauenswürdige Stelle eingerichtet werden könnte. Dazu wird kurz das Kryptosystem nach Paillier erläutert, um darauf aufbauend Möglichkeiten zur Lösung der gestellten Aufgabe zu zeigen.

2 Grundlagen

2.1 Kryptographische Grundlagen

Auf die kryptographischen Grundlagen und im Speziellen auf das RSA-System wird hier nicht eingegangen. Dazu wird auf die einschlägige Grundlagenliteratur zu Kryptographie verwiesen.

2.2 Paillier-Kryptosystem

Das Paillier-Kryptosystem basiert teilweise auf dem asymmetrischen RSA-System. Es muss ebenfalls ein Modulus N als Produkt von zwei sehr grossen Primzahlen p und q erzeugt werden. Im Folgenden wird das Paillier-Kryptosystem vorgestellt, weil es eine wesentliche Grundlage für dieses Paper ist.

2.2.1 Schlüsselerzeugung

Public Key

Der RSA-Modulus wird berechnet durch $N = p * q > T_{max}$ wobei T_{max}: Maximalzahl Votes (z.B. 5 Mio. in der Schweiz) und p, q: sichere, sehr grosse Primzahlen (safe primes: (p-1)/2 und (q-1)/2 sind auch Primzahlen) sowie $\gcd(N, \varphi(N) = (p-1) * (q-1)) = 1$. N und $\varphi(N)$ sind also teilerfremd zueinander.

Für den Generator G gilt folgende Bedingung:

$G \in Z^*_{N^2} \wedge \gcd(\eta, N) = 1$ wobei $\eta = \frac{(G^{\text{lcm}(p-1,q-1)} \bmod N^2) - 1}{N}$ und $Z^*_{N^2}$ die multiplikative Gruppe mod N^2 ist, welche nur die zu p und q teilerfremden Elemente enthält.

Meist wird die einfachste Möglichkeit für G gewählt, nämlich G = N + 1.

Private Key

Ein Teil des Private Keys wird berechnet durch $\lambda = \text{lcm}(p-1, q-1)$ und der andere Teil ist zu berechnen mit $\mu = \eta^{-1} \bmod N$ wobei η gleich berechnet wird wie oben. μ ist also das multiplikative Inverse von $\eta \bmod N$.

Zusammenfassung

Die beiden Schlüsselpaare sind demnach:

Public Key: (N, G)

Private Key: (μ, λ)

2.2.2 Verschlüsselung

Die Verschlüsselung funktioniert mit $\text{Encr}(M_i, r_i) = c_i = G^{M_i} * r_i^N \bmod N^2$, wobei M_i die Message, $r_i \in Z^*_N$ eine Zufallszahl zur Verschleierung der Stimme und Z^*_N die multiplikative Gruppe mod N ist.

2.2.3 Entschlüsselung

Die Entschlüsselung erfolgt mit $\text{Decr}(c_i) = M_i = \frac{(c_i^{\lambda} \bmod N^2) - 1}{N} * \mu \bmod N$.

Weitere Informationen zum Paillier-Kryptosystem finden sich unter [PaKr99] und [WiPa11].

2.3 Secret sharing und threshold decryption

Secret sharing und threshold decryption bedeutet, jede Partei hat ein Teilgeheimnis und die Kombination aller Teilgeheimnisse ergibt den Gesamtschlüssel, mit dem die Message entschlüsselt werden kann. Threshold bedeutet hier, dass es eine gewisse Mindestanzahl Teilnehmer t von allen n (klein N) beteiligten Parteien benötigt, um das Geheimnis zu entschlüsseln. Wenn die Anzahl Parteien diese Schwelle (threshold) unterschreitet, funktioniert das System nicht.

Die Notation für threshold decryption ist: (t, n), sprich: „t aus n (Parteien müssen kooperieren)“. Für die Mindestanzahl t muss ein Polynom f(x) vom Grad t - 1 generiert werden von der Form:

$$f(x) = \underbrace{s * x^0}_{\text{Geheimnis}} + a_1 * x^1 + a_2 * x^2 + ... + a_{t-1} * x^{t-1} \bmod N$$

Abbildung 1 zeigt ein Beispiel eines Polynoms 5-ten Grades mit der Auswertungsstelle i = 0 mit dem blauen Funktionswert f(0) = s, welcher dem Geheimnis entspricht und den 6 grünen Funktionswerten f(i) für Partei P_i.

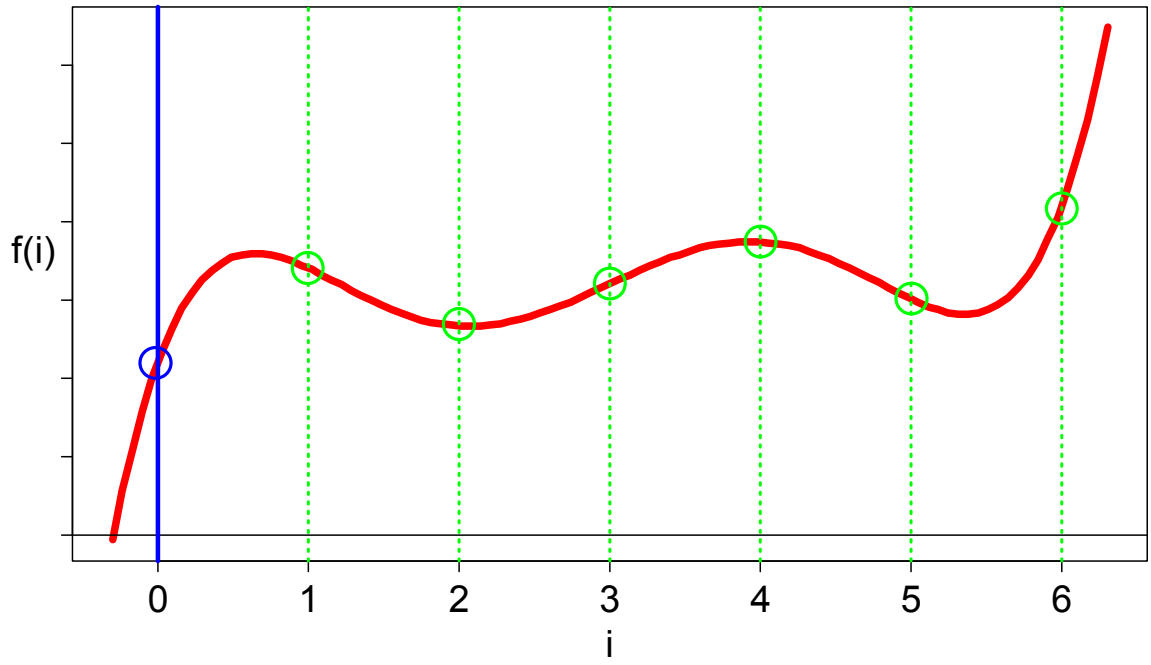

Abbildung 1: Beispiel eines Polynoms 5. Grades mit den verschiedenen Auswertungsstellen

Darüber hinaus verlangt secret sharing und threshold decryption einen so genannten *trusted Dealer*, der ebendieses Polynom inkl. des Geheimnisses und der Stützstellen generiert. Er erzeugt im Falle von Paillier das öffentliche und private Schlüsselpaar. Der trusted Dealer kann z.B. als sichere Hardware-Box implementiert sein, die, nachdem alle nötigen Parameter erzeugt wurden, „zerstört“ wird.

2.4 Without trusted Dealer

Without trusted dealer bedeutet, die Polynom- und Schlüsselerzeugung und die Ver- und Entschlüsselung sind verteilt. Es gibt also keine zentrale Stelle (trusted Dealer) mehr.

2.5 Probleme

Der trusted Dealer ist ein *single point of Attack*. Das heisst, wenn er kompromittiert ist, kann der Angreifer das ganze System beeinflussen. Darüber hinaus sind alle involvierten Parteien gezwungen, dem Dealer zu vertrauen (deshalb der Zusatz *trusted*).

Durch das Weglassen des Dealers ergeben sich aber andere Probleme. Nun müssen der RSA-Modulus N, die Schlüssel und die Polynome verteilt berechnet bzw. erzeugt werden. Speziell für N ergibt sich, dass keine der Parteien die Faktorisierung von N = p * q wissen darf bzw. errechnen kann.

3 Vorarbeiten

Zum Thema des verteilten Errechnens des RSA-Modulus N und gemeinsamen Berechnen von Geheimnissen wurden bereits einige Vorarbeiten publiziert.

3.1 Shoup

Victor Shoup zeigte 2000 mit seinem Paper „Practical Threshold Signatures" die Verwendung von RSA auf threshold decryption. Er schuf damit eine Basis für viele andere Untersuchungen auf diesem Gebiet. Sein System benötigt aber einen trusted Dealer.

3.2 Fouque & Stern

Pierre-Alain Fouque und Jacques Stern haben 2001 auf Basis von Shoup weitere Untersuchungen gemacht und diese unter „Fully Distributed Threshold RSA under Standard Assumptions“ publiziert. Sie haben gezeigt, dass das Protokoll von Shoup auch ohne trusted Dealer möglich ist, in dem sie dieses generalisierten.

3.3 Boneh & Franklin

Dan Boneh und Matthew Franklin zeigten 2001 in ihrem Paper „Efficient Generation of Shared RSA keys“ [BoFr01], wie die RSA-Parameter e_i, d_i und N erzeugt werden können, ohne dass die Faktorisierung $N = \left(\sum p_i\right) * \left(\sum q_i\right)$ irgendeiner Partei bekannt ist, so dass kein trusted Dealer mehr nötig ist.

3.3.1 BGW & Pedersen

Die Methode von Boneh & Franklin basiert teilweise auf den Arbeiten von Ben-Or, Goldwasser und Widgerson (BGW) [BGW88] und Pedersen [Ped91]. Diese beiden Methoden werden weiter unten ausführlicher beschrieben.

3.4 Nishide & Sakurai

Takashi Nishide und Kouichi Sakurai haben 2011 im Paper „Distributed Paillier Cryptosystem without Trusted Dealer“ [NiSa11] eine Lösung für ein komplett verteiltes Paillier-Kryptosystem gefunden.

Boneh & Franklin und Fouque & Stern sind für RSA konzipiert und funktionieren nicht direkt für Paillier, Nishide & Sakurai konnten aber dennoch einige Techniken übernehmen. Da deren Paper dem Thema dieses Berichts am nahesten kommt, wird es hier im Detail beschrieben.

4 Paillier-Kryptosystem nach Nishide & Sakurai *mit* Dealer

Weil beim Paper von Nishide & Sakurai die Erzeugung der Paillier-Schlüssel selbst *mit* trusted Dealer etwas vom von Pascal Paillier vorgeschlagenen Protokoll [PaKr99] abweicht und weil deren Methode *ohne* trusted Dealer auf ebendieser Methode aufbaut, ist auch diese hier beschrieben.

Das threshold decryption System sei (t, n). Zur Wiederholung: t ist die Mindestanzahl Parteien und n die Gesamtanzahl der teilnehmenden Parteien.

4.1 Schlüsselerzeugung

Die Schlüssel werden dann vom Dealer folgendermassen erzeugt:

1. $N = p * q \wedge \gcd(N, \varphi(N)) = 1$, wobei $p = 2 * p' + 1$ und $q = 2 * q' + 1$ mit p' und q' selbst Primzahlen. p und q sind also safe primes.

2. $\beta \in Z_N^*$ (zufällig gewählt), $m = p' * q'$, $\theta = m * \beta \bmod N$

3. Der $k_{pub}(N, G = N + 1, \theta)$ wird für die Verschlüsselung verwendet.

   Der $k_{priv}(\beta * m)$ weiss nur der Dealer. Er entspricht dem Geheimnis des secret sharings.

4. Der Dealer erzeugt das secret sharing Polynom von der Form:

   $f(x) = \beta * m * x^0 + a_1 * x^1 + a_2 * x^2 + ... + a_{t-1} * x^{t-1} \bmod N * m$

   Das Geheimnis ist folglich $f(0) = \beta * m$.

   $f(i) = \beta * m + a_1 * i + a_2 * i^2 + ... + a_{t-1} * i^{t-1} \bmod N * m$ wird an die Partei P_i mit i = [1…n] gesendet. Diese Werte sind dann nur der jeweiligen Partei P_i und dem Dealer bekannt.

Nebst diesen Schritten, gibt es auch noch Vorgänge für das Verifizieren der Keys. Diese sind hier aber nicht erläutert, sie können jedoch in [NiSa11] nachgelesen werden.

4.2 Verschlüsselung

Die Verschlüsselung kann von allen Parteien gemacht werden und erfolgt mit $c = G^M * x^N \bmod N^2$ wobei M die Message und $x \in Z_N^*$ (zufällig und absolut geheim von den einzelnen Parteien gewählt, um ihre Stimme zu verschleiern) ist.

4.3 Entschlüsselung

1. Partei P_i veröffentlicht ihr $c_i = c^{2*\Delta*f(i)} \bmod N^2$, genannt *partial decryption share*, wobei $\Delta = n!$.

2. Es existieren nun t partial decryption shares (es kann t = n sein, wenn alle involvierten Parteien kooperieren). Die ursprüngliche Message wird nun folgendermassen berechnet:

   $M = \left(\frac{c'-1}{N}\right) * (4 * \Delta^2 * \theta)^{-1} \bmod N$ wobei $c' = \prod c_i^{\lambda_i} \bmod N^2$.

   $\lambda_i = 2 * \Delta * \prod_{i \neq j} \frac{0 - j}{i - j}$ entsprechen den Lagrange-Koeffizienten (s. [WiLa11]). Die Lagrange-Koeffizienten werden mit $\Delta = n!$ (Fakultät) multipliziert, um sicherzustellen, dass der Term λ_i am Schluss durch alle natürlichen Zahlen bis und mit n (1, 2, 3, …, n) teilbar ist.

   Auf einen Beweis oder eine Herleitung wird an dieser Stelle verzichtet.

4.4 Rechenbeispiel

Um die genannte Methode zu demonstrieren, wird nachfolgend ein Zahlenbeispiel mit n = 4 Parteien, einem Dealer und einer Mindestanzahl Parteien von t = 3 durchgerechnet. Der Übersicht halber sind die Zahlen kleiner als in einem echten E-Voting-System gewählt.

4.4.1 Schlüsselerzeugung

Die grundlegenden Variablen seien:

$$p' = 3 \Rightarrow p = 2 * 3 + 1 = 7$$
$$q' = 5 \Rightarrow q = 2 * 5 + 1 = 11$$
$$N = p * q = 7 * 11 = 77$$
$$\varphi(N) = (p - 1) * (q - 1) = (7 - 1) * (11 - 1) = 60 = \varphi$$
$$\beta \in Z_N^* = 8$$
$$m = p' * q' = 3 * 5 = 15$$
$$\theta = m * \beta \bmod N = 15 * 8 \bmod 77 = 43$$

Die beiden Schlüsselpaare sind demnach:

$k_{pub}(N, G = N + 1, \theta) = (77, 78, 43)$

$k_{priv}(\beta * m = 8 * 15) = (120)$

4.4.2 Verschlüsselung

Der Plain- und Ciphertext der Message wird wie folgt erzeugt:

$$M = 16$$
$$c = G^M * x^N \bmod N^2 \mid x \in Z_N^* = 10$$
$$c = 78^{16} * 10^{77} \bmod 77^2 = \underline{1671}$$
$$n = 4 \text{ Parteien}, t = 3, t - 1 = 2, \Delta = n! = 4! = 24$$

4.4.3 Erzeugung des Polynoms und der partial decryption shares

Das Polynom 2-ten Grades (t - 1 = 2), die Funktionswerte dazu und die einzelnen partial decryption shares werden folgendermassen gewählt:

$$f(x) = \beta * m + a_1 * x + a_2 * x^2 \bmod N * m$$
$$= 120 + 69 * x + 333 * x^2 \bmod (77 * 15) \qquad c_i = c^{2*\Delta*f(i)} \bmod N^2$$
$$f(1) = 120 + 69 * 1 + 333 * 1^2 \bmod 1155 = 522 \Rightarrow c_1 = c^{2*\Delta*f(1)} \bmod N^2 = 1671^{2*24*522} \bmod 77^2 = \underline{2619}$$
$$f(2) = 120 + 69 * 2 + 333 * 2^2 \bmod 1155 = 435 \Rightarrow c_2 = c^{2*\Delta*f(2)} \bmod N^2 = 1671^{2*24*435} \bmod 77^2 = \underline{4159}$$
$$f(3) = 120 + 69 * 3 + 333 * 3^2 \bmod 1155 = 1014 \Rightarrow c_3 = c^{2*\Delta*f(3)} \bmod N^2 = 1671^{2*24*1014} \bmod 77^2 = \underline{3928}$$
$$f(4) = 120 + 69 * 4 + 333 * 4^2 \bmod 1155 = 1104 \Rightarrow c_4 = c^{2*\Delta*f(4)} \bmod N^2 = 1671^{2*24*1104} \bmod 77^2 = \underline{1926}$$

4.4.4 Lagrange-Interpolation

Die Parameter für die Lagrange-Interpolation sind unabhängig von M oder c, sondern nur abhängig von der Anzahl Parteien n und t. Bei der Berechnung dieser ist auf die Vorzeichen zu achten:

$$\lambda_i = 2 * \Delta * \prod_{i \neq j} \frac{0 - j}{i - j}$$

$$\lambda_1 = 2 * \Delta * \frac{0-2}{1-2} * \frac{0-3}{1-3} * \frac{0-4}{1-4} = 2 * \Delta * 4$$
$$\lambda_2 = 2 * \Delta * \frac{0-1}{2-1} * \frac{0-3}{2-3} * \frac{0-4}{2-4} = 2 * \Delta * (-6)$$
$$\lambda_3 = 2 * \Delta * \frac{0-1}{3-1} * \frac{0-2}{3-2} * \frac{0-4}{3-4} = 2 * \Delta * 4$$
$$\lambda_4 = 2 * \Delta * \frac{0-1}{4-1} * \frac{0-2}{4-2} * \frac{0-3}{4-3} = 2 * \Delta * (-1)$$

Unter der Annahme, dass nur t = 3 Parteien mitgemacht haben (z.B. Partei 3 ist ausgestiegen), können wir auch folgendes berechnen:

$$\lambda_1 = 2 * \Delta * \frac{0-2}{1-2} * \frac{0-4}{1-4} = 128$$
$$\lambda_2 = 2 * \Delta * \frac{0-1}{2-1} * \frac{0-4}{2-4} = -96$$
$$\lambda_4 = 2 * \Delta * \frac{0-1}{4-1} * \frac{0-2}{4-2} = 16$$

4.4.5 Vorarbeit zur Entschlüsselung

$$c' = \prod c_i^{\lambda_i} \bmod N^2$$
$$= (2619^{2*24*4} * 4159^{2*24*(-6)} * 3928^{2*24*4} * 1926^{2*24*(-1)}) \bmod 77^2$$
$$= (\quad 4621 \quad * \quad 155 \quad * \quad 1003 \quad * \quad 2465 \quad) \bmod 77^2$$
$$= \underline{2311}$$

Wenn z.B. Partei 3 ausgestiegen ist, führt uns das zum gleichen Resultat:

$$c' = (2619^{128} * 4159^{-96} * 1926^{16}) \bmod 77^2$$
$$= (\quad 3081 \quad * \quad 4005 \quad * \quad 1156 \quad) \bmod 77^2$$
$$= \underline{2311}$$

4.4.6 Entschlüsselung

Die ursprüngliche Message M = 16 kann nun folgendermassen entschlüsselt werden:

$$M = \left(\frac{c'-1}{N}\right) * (4 * \Delta^2 * \theta)^{-1} \bmod N = \underbrace{\left(\frac{2311-1}{77}\right)}_{30} * \underbrace{(4 * 24^2 * 43)^{-1}}_{57\ (EEA)} \bmod 77 = \underline{\underline{16}}$$ ✓

4.4.7 Übersicht über die erzeugten Variabeln

Dealer & public	Dealer & private	Partei P_i & private	Partei P_i & public
N, G, t, n, λ_i, Δ, θ, c'	p, q, p', q', β, m, f(x), f(i), f(0) = β * m	M, f(i)	c, c_i

5 Komplett verteiltes Paillier-Kryptosystem ohne trusted Dealer

Nachfolgend wird ein komplettes, verteiltes Paillier-Kryptosystem erläutert, das auf [NiSa11], [BoFr01], [BGW88] und [Ped91] basiert.

5.1 Schlüsselerzeugung

Die Schlüsselerzeugung ist nachfolgend gezeigt:

1. Über den standardisierten, verteilten Algorithmus BGW, welcher in [BoFr01] beschrieben ist berechnen die Parteien den RSA-Modulus N = p * q, so dass weder $p = \left(\sum p_i\right)$ noch $q = \left(\sum q_i\right)$ mit i = [1…n] für Partei P_i bekannt werden.

2. Partei P_i erzeugt $\beta_i \in [0, K * N]$ und $R_i \in [0, K^2 * N]$ zufällig. Heute wird $K \approx 2^{128}$ gewählt.

   Es wird nun ein Teil des Public Keys $\theta' = \Delta * \varphi * \left(\sum \beta_i\right) + N * \left(\sum \Delta * R_i\right)$ berechnet, wobei $\Delta * R_i$ *über die Menge der ganzen Zahlen verteilt erzeugt* wird, weil $m = p' * q'$ für $f(x) \bmod N * m$ unbekannt ist. Die Summe $\beta = \sum \beta_i$ und auch $\varphi = (p-1) * (q-1)$ sind ebenfalls unbekannt. Das Produkt $\varphi * \beta$ muss demnach verteilt erzeugt werden, ohne dass jemand φ oder β kennt. Weil $\varphi = \left(\left(\sum p_i\right) - 1\right) * \left(\left(\sum q_i\right) - 1\right) = \left(\sum p_i\right) * \left(\sum q_i\right) - \left(\sum p_i\right) - \left(\sum q_i\right) + 1$ gilt $\varphi = N + 1 - \sum (p_i + q_i)$, deshalb ist $\varphi * \beta = \left(N + 1 - \sum (p_i + q_i)\right) * \left(\sum \beta_i\right)$. Diese Berechnung kann mit Hilfe vom BGW-Protokoll analog zur Berechnung vom Produkt $N = p * q$ durchgeführt werden, ist aber im Paper von Nishide & Sakurai nicht näher definiert.

   Der Public Key besteht nun aus: $k_{pub}(N, G = N + 1, \theta' = \Delta * \varphi * \beta \bmod N)$.

   Anmerkung a. Beim System *mit* Dealer ist $\theta = m * \beta \bmod N$, also ein m-faches von β. Weil $m = p' * q'$, $p' = (p - 1)/2$ und $q' = (q - 1)/2$, ergibt sich $4m = (p-1) * (q-1) = \varphi$. Deshalb ist $\theta' \sim 4m * \beta$ wiederum ein m-faches (dargestellt durch $\varphi/4$) von β. Freilich sind hier weder m noch φ bekannt.

3. Die $\Delta * R_i$ werden nun über der Menge der Ganzzahlen (engl.: integers, d.h. kein Modulus) mit einer modifizierten Version des Verfahrens von Pedersen [Ped91] verteilt erzeugt. Es basiert im Wesentlichen darauf, dass mehrere geheime, zufällige, aber speziell gewählte Polynome der Parteien an gewissen Stellen $s_j(i)$ summiert werden und das Summenpolynom $f_R(x)$ dann an der Nullstelle $f_R(0)$ ausgewertet wird. Das schlussendlich verwendete Polynom f(x) ist dann folgendermassen zu berechnen:

   $f(x) = N * f_R(x) - \theta'$

Anmerkung b. Beim System *ohne* Dealer kommt bei der Definition von θ' im Vergleich zum System *mit* Dealer der Summand $N * \left(\sum \Delta * R_i\right)$ hinzu. Wenn man diese Definition umformt und das mit der Definition von f(x) von oben vergleicht, sieht man:

$$-\Delta\varphi\beta = N * \left(\sum \Delta * R_i\right) - \theta'$$
$$f(x) = N * f_R(x) - \theta'$$

5.2 Verschlüsselung

Die Verschlüsselung erfolgt wie *mit* trusted Dealer mit $c = G^M * x^N \bmod N^2$.

5.3 Entschlüsselung

Die Entschlüsselungsphase erfolgt ähnlich wie mit einem Dealer, ausser dem Minuszeichen in M.

1. Partei P_i veröffentlicht ihr $c_i = c^{2*\Delta*f(i)} \bmod N^2$.
2. $M = \left(\frac{c'-1}{N}\right) * (-4 * \Delta^2 * \theta')^{-1} \bmod N$ wobei $c' = \prod c_i^{\lambda_i} \bmod N^2$ und $\lambda_i = 2 * \Delta * \prod_{i \neq j} \frac{0-j}{i-j}$.

5.4 Unterschied zu *mit* Dealer

Da kein trusted Dealer vorhanden ist, müssen alle Parameter von den Parteien im Einklang mit den anderen Parteien erzeugt werden. p_i, q_i, $s_i(x)$ und β_i sind für jede Partei geheim und nicht mehr nur einer zentralen Stelle bekannt. Der $k_{pub}(N, G = N+1, \theta')$ und das Summenpolynom f(x) wurden verteilt erzeugt. f(x) wurde über den Ganzzahlen und nicht über einem endlichen Körper verteilt erzeugt.

5.5 Rechenbeispiel

5.5.1 Verteiltes Berechnen des Modulus N nach BGW

Die Erzeugung der Schlüssel und Polynome geschieht als Vorlauf der Wahlen, weil es bei grossen Zahlen relativ viel Zeit und Aufwand in Anspruch nimmt.

Zuerst muss $N = \left(\sum p_i\right) * \left(\sum q_i\right)$ nach **BGW** aus [BoFr01] berechnet werden. Dazu wählen alle Parteien P_i geheim die beiden Primzahlen p_i und q_i:

$P_1: p_1 = 2, \quad q_1 = 3$
$P_2: p_2 = 2, \quad q_2 = 7$
$P_3: p_3 = 2, \quad q_3 = 11$
$P_4: p_4 = 5 \quad q_4 = 2$

Polynome und Funktionswerte erzeugen

Ausserdem einigen sie sich auf eine Primzahl P, welche so gewählt sein muss, dass sie sicher grösser als N sein wird. Es sei P = 257. Jede Partei P_i muss nun 3 Polynome der Form

$f_i(x) = p_i + a_1 * x^1 + ... + a_u * x^u \bmod P,$
$g_i(x) = q_i + a_1 * x^1 + ... + a_u * x^u \bmod P$ und
$h_i(x) = 0 + a_1 * x^1 + ... + a_{2u} * x^{2u} \bmod P$

erzeugen und für sich behalten, wobei $u \le (n-1)/2$, hier also $u = 1$:

P_1:	$f_1(x) = 2 + 7x \bmod P$	$g_1(x) = 3 + 37x \bmod P$	$h_1(x) = 0 + 13x + 53x^2 \bmod P$
P_2:	$f_2(x) = 2 + 15x \bmod P$	$g_2(x) = 7 + 9x \bmod P$	$h_2(x) = 0 + 10x + 11x^2 \bmod P$
P_3:	$f_3(x) = 2 + 6x \bmod P$	$g_3(x) = 11 + 12x \bmod P$	$h_3(x) = 0 + 3x + 2x^2 \bmod P$
P_4:	$f_4(x) = 5 + 8x \bmod P$	$g_4(x) = 2 + 3x \bmod P$	$h_4(x) = 0 + 9x + 10x^2 \bmod P$

Nun sendet jede Partei P_i einer anderen Partei P_j die Werte $p_{i,j} = f_i(j)$, $q_{i,j} = g_i(j)$ und $h_{i,j} = h_i(j)$.

P_1:	$p_{1,1} = f_1(1) = 9$	$q_{1,1} = g_1(1) = 40$	P_2:	$p_{1,2} = f_1(2) = 16$	$q_{1,2} = g_1(2) = 77$
	$p_{2,1} = f_2(1) = 17$	$q_{2,1} = g_2(1) = 16$		$p_{2,2} = f_2(2) = 32$	$q_{2,2} = g_2(2) = 25$
	$p_{3,1} = f_3(1) = 8$	$q_{3,1} = g_3(1) = 23$		$p_{3,2} = f_3(2) = 14$	$q_{3,2} = g_3(2) = 35$
	$p_{4,1} = f_4(1) = 13$	$q_{4,1} = g_4(1) = 5$		$p_{4,2} = f_4(2) = 21$	$q_{4,2} = g_4(2) = 8$
	$h_{1,1} = h_1(1) = 66$	$h_{2,1} = h_2(1) = 21$		$h_{1,2} = h_1(2) = 238$	$h_{2,2} = h_2(2) = 64$
	$h_{3,1} = h_3(1) = 5$	$h_{4,1} = h_4(1) = 19$		$h_{3,2} = h_3(2) = 14$	$h_{4,2} = h_4(2) = 58$
P_3:	$p_{1,3} = f_1(3) = 23$	$q_{1,3} = g_1(3) = 114$	P_4:	$p_{1,4} = f_1(4) = 30$	$q_{1,4} = g_1(4) = 151$
	$p_{2,3} = f_2(3) = 47$	$q_{2,3} = g_2(3) = 34$		$p_{2,4} = f_2(4) = 62$	$q_{2,4} = g_2(4) = 43$
	$p_{3,3} = f_3(3) = 20$	$q_{3,3} = g_3(3) = 47$		$p_{3,4} = f_3(4) = 26$	$q_{3,4} = g_3(4) = 59$
	$p_{4,3} = f_4(3) = 29$	$q_{4,3} = g_4(3) = 11$		$p_{4,4} = f_4(4) = 37$	$q_{4,4} = g_4(4) = 14$
	$h_{1,3} = h_1(3) = 2$	$h_{2,3} = h_2(3) = 129$		$h_{1,4} = h_1(4) = 129$	$h_{2,4} = h_2(4) = 216$
	$h_{3,3} = h_3(3) = 27$	$h_{4,3} = h_4(3) = 117$		$h_{3,4} = h_3(4) = 44$	$h_{4,4} = h_4(4) = 196$

Teil-Moduli N_i berechnen

Jede Partei P_i berechnet und veröffentlicht nun einen Teil-Modulus N_i, welcher einem Stützwert des Gesamtpolynoms entspricht:

$$N_i = \left(\sum p_{i,j}\right) * \left(\sum q_{i,j}\right) + \left(\sum h_{i,j}\right) \bmod P$$

P_1:	$N_1 = (9 + 17 + 8 + 13) * (40 + 16 + 23 + 5) + (66 + 21 + 5 + 19) \bmod 257 =$	204
P_2:	$N_2 = (16 + 32 + 14 + 21) * (77 + 25 + 35 + 8) + (238 + 64 + 14 + 58) \bmod 257 =$	73
P_3:	$N_3 = (23 + 47 + 20 + 29) * (114 + 34 + 47 + 11) + (2 + 129 + 27 + 117) \bmod 257 =$	117
P_4:	$N_4 = (30 + 62 + 26 + 37) * (151 + 43 + 59 + 14) + (129 + 216 + 44 + 196) \bmod 257 =$	79

Bestimmen und verifizieren von $N = \Sigma p_i * \Sigma q_i$

Auf diese Stützwerte wird nun die Lagrange-Interpolation angewendet:

$$N = 204 * 4 + 73 * (-6) + 117 * 4 + 79 * (-1) \equiv \underline{253} \bmod 257$$

Aus der Definition von weiter oben lässt sich dieser Wert theoretisch verifizieren:

$$N = \left(\sum p_i\right) * \left(\sum q_i\right) = (2 + 2 + 2 + 5) * (3 + 7 + 11 + 2) = 11 * 23 = \underline{253}$$ ✓

An dieser Stelle muss mit dem unter [BoFr01] beschriebenen Verfahren *trial division* getestet werden, ob N wirklich das Produkt von zwei Primzahlen p und q ist. Die einzelnen Primzahlen p_i und q_i sind hier bewusst so gewählt worden, dass p und q kleine, aber sichere Primzahlen sind. In [NiSa11] sind einige weitere Bedingungen für p_i und q_i aufgeführt. Im Allgemeinen sollten sie durch 4 teilbar sein mit Ausnahme der Partei P_1. Dort soll gelten $p_1 \equiv q_1 \equiv 3 \bmod 4$. Diese Regel dient aber nur dazu, dass man eine möglichst grosse Chance hat, zwei Primzahlen als $\sum p_i$ und $\sum q_i$ zu erhalten.

5.5.2 Verteiltes Berechnen des Produktes φ * β nach BGW

Für den Public Key θ' brauchen wir ausser N und Δ auch noch $\varphi * \beta = \left(N + 1 - \sum (p_i + q_i)\right) * \left(\sum \beta_i\right)$ und $\left(\sum \Delta * R_i\right)$.

Die Faktoren des Produkts $\varphi * \beta$ dürfen nicht bekannt sein, wohl aber das Ergebnis. Dieses wird analog zu N = p * q mit **BGW** berechnet.

φ lässt sich umformen in

$$\varphi = \underbrace{(N + 1 - (p_1 + q_1))}_{\text{Summand } r_1} \underbrace{-(p_2 + q_2)}_{r_2} \underbrace{-(p_3 + q_3)}_{r_3} \underbrace{-(p_4 + q_4)}_{r_4}.$$

Diese Summanden r_i sind nachfolgend samt den von den Parteien zufällig gewählten und geheimen β_i aufgelistet (P = 257 wurde von oben übernommen, wobei das nicht immer möglich ist, da die β_i per Definition viel grösser als N sein können und P hier nur knapp grösser als N ist):

$P_1 : r_1 = N + 1 - (p_1 + q_1) = 253 + 1 - (2 + 3) \equiv \underline{249} \bmod 257$ $\quad \beta_1 = 10$
$P_2 : r_2 = -(p_2 + q_2) = -(2 + 7) = -9 \equiv \underline{248} \bmod 257$ $\quad \beta_2 = 37$
$P_3 : r_3 = -(p_3 + q_3) = -(2 + 11) = -13 \equiv \underline{244} \bmod 257$ $\quad \beta_3 = 15$
$P_4 : r_4 = -(p_4 + q_4) = -(5 + 2) = -7 \equiv \underline{250} \bmod 257$ $\quad \beta_4 = 14$

Polynome und Funktionswerte erzeugen

Jede Partei P_i muss nun wiederum 3 Polynome der Form

$f_i(x) = r_i + a_1 * x^1 + ... + a_u * x^u \bmod P,$
$g_i(x) = \beta_i + a_1 * x^1 + ... + a_u * x^u \bmod P$ und
$h_i(x) = 0 + a_1 * x^1 + ... + a_{2u} * x^{2u} \bmod P$

erzeugen und für sich behalten:

P_1: $f_1(x) = 249 + 7x \bmod P$ $\quad g_1(x) = 10 + 11x \bmod P$ $\quad h_1(x) = 0 + 13x + 53x^2 \bmod P$
P_2: $f_2(x) = 248 + 15x \bmod P$ $\quad g_2(x) = 37 + 9x \bmod P$ $\quad h_2(x) = 0 + 10x + 11x^2 \bmod P$
P_3: $f_3(x) = 244 + 6x \bmod P$ $\quad g_3(x) = 15 + 12x \bmod P$ $\quad h_3(x) = 0 + 3x + 2x^2 \bmod P$
P_4: $f_4(x) = 250 + 8x \bmod P$ $\quad g_4(x) = 14 + 3x \bmod P$ $\quad h_4(x) = 0 + 9x + 10x^2 \bmod P$

Nun sendet jede Partei P_i einer anderen Partei P_j die Werte $r_{i,j} = f_i(j)$, $\beta_{i,j} = g_i(j)$ und $h_{i,j} = h_i(j)$:

P_1:	$r_{1,1} = f_1(1) = 256$	$\beta_{1,1} = g_1(1) = 21$	P_2	$r_{1,2} = f_1(2) = 6$	$\beta_{1,2} = g_1(2) = 32$
	$r_{2,1} = f_2(1) = 6$	$\beta_{2,1} = g_2(1) = 46$		$r_{2,2} = f_2(2) = 21$	$\beta_{2,2} = g_2(2) = 55$
	$r_{3,1} = f_3(1) = 250$	$\beta_{3,1} = g_3(1) = 27$		$r_{3,2} = f_3(2) = 256$	$\beta_{3,2} = g_3(2) = 39$
	$r_{4,1} = f_4(1) = 9$	$\beta_{4,1} = g_4(1) = 17$		$r_{4,2} = f_4(2) = 9$	$\beta_{4,2} = g_4(2) = 20$
	$h_{1,1} = h_1(1) = 66$	$h_{2,1} = h_2(1) = 21$		$h_{1,2} = h_1(2) = 238$	$h_{2,2} = h_2(2) = 64$
	$h_{3,1} = h_3(1) = 5$	$h_{4,1} = h_4(1) = 19$		$h_{3,2} = h_3(2) = 14$	$h_{4,2} = h_4(2) = 58$
P_3:	$r_{1,3} = f_1(3) = 13$	$\beta_{1,3} = g_1(3) = 43$	P_4:	$r_{1,4} = f_1(4) = 20$	$\beta_{1,4} = g_1(4) = 54$
	$r_{2,3} = f_2(3) = 36$	$\beta_{2,3} = g_2(3) = 64$		$r_{2,4} = f_2(4) = 51$	$\beta_{2,4} = g_2(4) = 73$
	$r_{3,3} = f_3(3) = 5$	$\beta_{3,3} = g_3(3) = 51$		$r_{3,4} = f_3(4) = 11$	$\beta_{3,4} = g_3(4) = 63$
	$r_{4,3} = f_4(3) = 17$	$\beta_{4,3} = g_4(3) = 23$		$r_{4,4} = f_4(4) = 25$	$\beta_{4,4} = g_4(4) = 26$
	$h_{1,3} = h_1(3) = 2$	$h_{2,3} = h_2(3) = 129$		$h_{1,4} = h_1(4) = 129$	$h_{2,4} = h_2(4) = 216$
	$h_{3,3} = h_3(3) = 27$	$h_{4,3} = h_4(3) = 117$		$h_{3,4} = h_3(4) = 44$	$h_{4,4} = h_4(4) = 196$

Teil-Produkte Ω_i berechnen

Jede Partei P_i berechnet und veröffentlicht nun ein Teil-Produkt Ω_i, welches einem Stützwert des Gesamtpolynoms entspricht:

$$\Omega_i = \left(\sum r_{i,j}\right) * \left(\sum \beta_{i,j}\right) + \left(\sum h_{i,j}\right) \bmod P$$

$$P_1: \quad \Omega_1 = (256 + 6 + 250 + 9) * (21 + 46 + 27 + 17) + (66 + 21 + 5 + 19) \bmod 257 = \quad 0$$
$$P_2: \quad \Omega_2 = (6 + 21 + 256 + 9) * (32 + 55 + 39 + 20) + (238 + 64 + 14 + 58) \bmod 257 = \quad 87$$
$$P_3: \quad \Omega_3 = (13 + 36 + 5 + 17) * (43 + 64 + 51 + 23) + (2 + 129 + 27 + 117) \bmod 257 = \quad 19$$
$$P_4: \quad \Omega_4 = (20 + 51 + 11 + 25) * (54 + 73 + 63 + 26) + (129 + 216 + 44 + 196) \bmod 257 = \quad 53$$

Bestimmen und verifizieren von $\varphi * \beta$

Auf diese Stützwerte wird nun die Lagrange-Interpolation angewendet:

$$\varphi * \beta = 0 * 4 + 87 * (-6) + 19 * 4 + 53 * (-1) \equiv \underline{15} \bmod 257$$

Aus der Definition von weiter oben lässt sich dieser Wert theoretisch verifizieren:

$$\varphi * \beta = \varphi(253) * \left(\sum \beta_i\right) = (11 - 1) * (23 - 1) * (10 + 37 + 15 + 14) \equiv \underline{15} \bmod 257 \quad \checkmark$$

5.5.3 Verteiltes Berechnen von $R = \Sigma R_i$ mit Pedersen

Die Nullstelle des für die Vorbereitung der Verschlüsselung benötigten Polynoms $f_R(0)$ und $\sum \Delta * R_i$ werden nach **Pedersen** errechnet.

Polynome und Funktionswerte erzeugen

Jede Partei P_i wählt für sich R_i und ein Polynom $s_i(x)$, so dass $s_i(0) = \Delta * R_i$:

$P_1 : R_1 = 9, \quad s_1(x) = \Delta * 9 + 10 * x + 9 * x^2$
$P_2 : R_2 = 16, \quad s_2(x) = \Delta * 16 + 2 * x + 3 * x^2$
$P_3 : R_3 = 3, \quad s_3(x) = \Delta * 3 + 3 * x + 2 * x^2$
$P_4 : R_4 = 7, \quad s_4(x) = \Delta * 7 + 4 * x + 2 * x^2$

Zwischenergebnisse $s_i(j)$ berechnen

Die Partei P_i berechnet nun die Werte für $s_i(j)$ mit j = [1…4]. Alle $s_i(j)$-Werte ausser $s_i(i)$ werden nun an die entsprechenden Parteien P_j gesendet.

$\Rightarrow s_1(1) = 235,\ s_1(2) = 272, s_1(3) = 327, s_1(4) = 400$
$\Rightarrow s_2(1) = 389,\ s_2(2) = 400, s_2(3) = 417, s_2(4) = 440$
$\Rightarrow s_3(1) = 77,\ s_3(2) = 86,\ s_3(3) = 99,\ s_3(4) = 116$
$\Rightarrow s_4(1) = 174,\ s_4(2) = 184, s_4(3) = 198, s_4(4) = 216$

$s_i(i)$: Partei behält Wert für sich

Bestimmen und verifizieren von R

Partei P_i summiert alle erhaltenen Werte $s_j(i)$ und ihr eigenes $s_i(i)$. Das Ergebnis jeder Partei wird mit den Lagrange-Koeffizienten multipliziert und diese Zwischenergebnisse werden dann aufsummiert. Dieser Wert entspricht $\sum \Delta * R_i$ und ist öffentlich, ohne dass die Parteien die einzelnen Summanden von $f_R(i)$ bekannt gegeben haben:

P_1 :	$f_R(1) = 235 + 389 + 77 + 174 =$ 875	$\Rightarrow 875 * 4$	3500
P_2 :	$f_R(2) = 272 + 400 + 86 + 184 =$ 942	$\Rightarrow 942 * (-6)$	- 5652
P_3 :	$f_R(3) = 327 + 417 + 99 + 198 =$ 1041	$\Rightarrow 1041 * 4$	4164
P_4 :	$f_R(4) = 400 + 440 + 116 + 216 =$ 1172	$\Rightarrow 1172 * (-1)$	- 1172
		$\Sigma =$	840

Aus der Definition von weiter oben lässt sich dieser Wert theoretisch verifizieren:

$R = \sum(\Delta * R_i) = \Delta * 9 + \Delta * 16 + \Delta * 3 + \Delta * 7 = 216 + 384 + 72 + 168 = 840$ ✓

Grafisch ist sich dieses Verfahren in Abbildung 2 veranschaulicht. Grün sind dabei die öffentlichen Funktionswerte $f_R(i)$ und blau $f_R(0) = R = 875 * 4 + 942 * (-6) + 1041 * 4 + 1172 * (-1) = 840$.

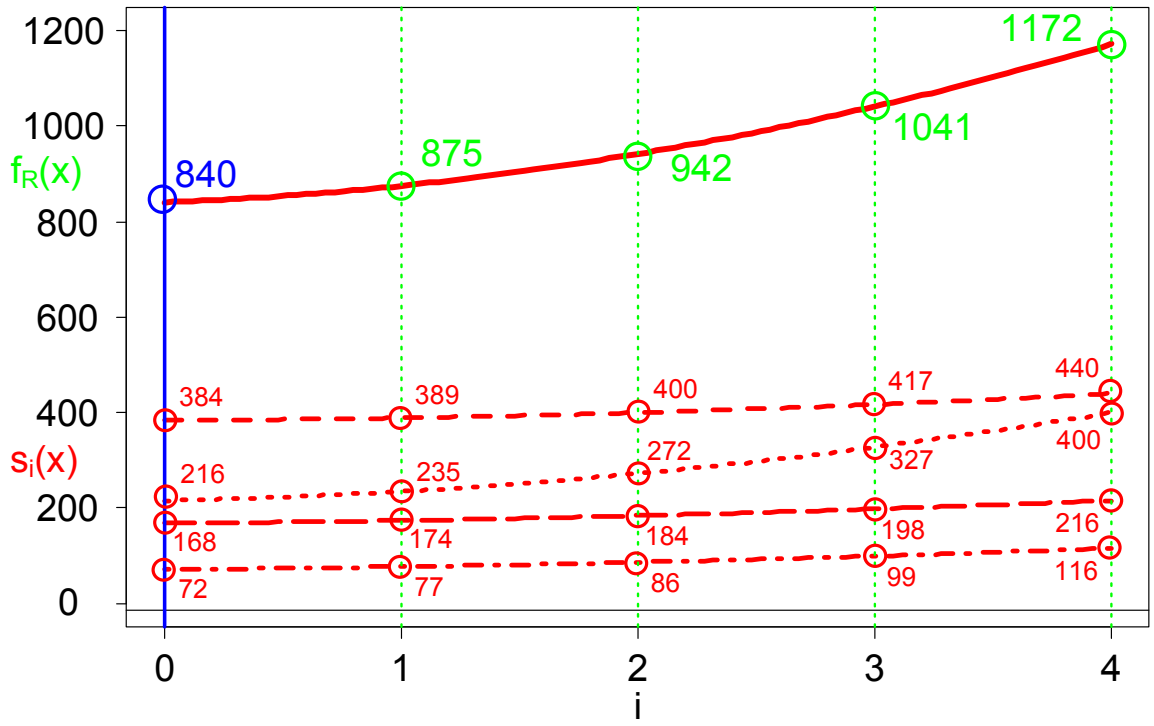

Abbildung 2: Summenpolynom nach dem Pedersen-Verfahren

5.5.4 Berechnen des Public Keys

θ' wird nun folgendermassen berechnet:

$$\theta' = \Delta * \underbrace{\varphi * \left(\sum \beta_i\right)}_{15} + N * \underbrace{\left(\sum \Delta * R_i\right)}_{840} = 24 * 15 + 253 * 840 = \underline{212'880}$$

5.5.5 Berechnen des Verschlüsselungspolynomstützstellen

Da wir $f(i) = N * f_R(i) - \theta'$ für die Teilentschlüsselung brauchen, berechnen wir:

$f_R(1) = 875 \quad \Rightarrow f(1) = 253 * 875 - 212'880 = \underline{8495}$
$f_R(2) = 942 \quad \Rightarrow f(2) = 253 * 942 - 212'880 = \underline{25'446}$
$f_R(3) = 1041 \quad \Rightarrow f(3) = 253 * 1041 - 212'880 = \underline{50'493}$
$f_R(4) = 1172 \quad \Rightarrow f(4) = 253 * 1172 - 212'880 = \underline{83'636}$

Wir befinden uns nun für $f(i)$ und $f_R(i)$ per Definition in den Ganzzahlen und nicht mehr in einem endlichen Körper, deshalb können die Werte beliebig gross sein.

5.5.6 Verschlüsselung

Die Verschlüsselung muss in einem realen E-Voting-System im Gegensatz zur Berechnung der Keys in Echtzeit geschehen.

Der Plain- und Ciphertext der Message sei:

$M = 156$

$c = G^M * x^N \bmod N^2 \mid x \in Z_N^* = 63$

$c = 254^{156} * 63^{253} \bmod 253^2 = \underline{57'338}$

$n = 4$ Parteien, $t = 3, t - 1 = 2, \Delta = n! = 4! = 24$

$f(1) = 8495 \Rightarrow c_1 = c^{2*\Delta*f(1)} \bmod N^2 = 57'338^{2*24*8495} \bmod 253^2 = \underline{8856}$

$f(2) = 25'446 \Rightarrow c_2 = c^{2*\Delta*f(2)} \bmod N^2 = 57'338^{2*24*25'446} \bmod 253^2 = \underline{16'262}$

$f(3) = 50'493 \Rightarrow c_3 = c^{2*\Delta*f(3)} \bmod N^2 = 57'338^{2*24*50'493} \bmod 253^2 = \underline{48'002}$

$f(4) = 83'636 \Rightarrow c_4 = c^{2*\Delta*f(4)} \bmod N^2 = 57'338^{2*24*83'636} \bmod 253^2 = \underline{16'262}$

5.5.7 Vorarbeit zur Entschlüsselung

$$c' = \prod c_i^{\lambda_i} \bmod N^2$$
$$= (8856^{2*24*4} * 16'262^{2*24*(-6)} * 48'002^{2*24*4} * 16'262^{2*24*(-1)}) \bmod 253^2$$
$$= (\ 35'927 * 29'694 * 54'971 * 13'502\ ) \bmod 253^2$$
$$= \underline{40'987}$$

Wenn wiederum nur t = 3 Parteien mitgemacht haben (z.B. Partei 3 ist ausgestiegen), erhalten wir dasselbe Resultat:

$$c' = (8856^{128} * 16'262^{-96} * 16'262^{16}) \bmod 253^2$$
$$= (\ 45'288 * 6372 * 32'707\ ) \bmod 253^2$$
$$= \underline{40'987}$$

5.5.8 Entschlüsselung

Die ursprüngliche Message M = 156 kann nun folgendermassen entschlüsselt werden:

$$M = \left(\frac{c'-1}{N}\right) * (-4 * \Delta^2 * \theta')^{-1} \bmod N = \underbrace{\left(\frac{40'987 - 1}{253}\right)}_{162} * \underbrace{(-4 * 24^2 * 212'880)^{-1}}_{179\ (EEA)} \bmod 253 = \underline{\underline{156}}$$ ✓

5.5.9 Übersicht über die erzeugten Variabeln

Public	Unbekannt	Partei-private
N, G, n, t, Δ, θ', $\sum \Delta * R_i$, $f_R(i)$, f(i), c_i, λ_i, c', N_i, Ω_i. φ * β	p, q, φ, $\sum \beta_i$	p_i, q_i, r_i, R_i, β_i, $f_i(x)$, $g_i(x)$, $h_i(x)$, $s_i(x)$, $s_j(i)$, $s_i(i)$

6 Zusammenfassung

Dieses Paper untersucht mehrere verteilte Verfahren der Schlüsselerzeugung für RSA und die Versuche, diese auf das Paillier-Kryptosystem anzuwenden.

Ein Paillier-Kryptosystem mit verteilter Erzeugung der Schlüssel und verteilten Berechnungen von Geheimnis und Klartexten ist dank den Papern von Takashi Nishide & Kouichi Sakurai und Dan Boneh & Matthew Franklin zwar sehr aufwendig, aber lösbar. Die Idee beruht darauf, dass einige Polynome und Geheimnisse über den Ganzzahlen anstatt in einem endlichen Körper nach Pedersen und der RSA-Modulus N nach dem Verfahren von BGW verteilt erzeugt werden.

7 Glossar

BGW	Protokoll von **B**en-Or, **G**oldwasser und **W**idgerson zum verteilten Berechnen nach [BGW88]
EEA	**E**rweiterter **e**uklidischer **A**lgorithmus zum Finden des gcd und modularen Inversen.
gcd	**g**reatest **c**ommon **d**ivisor (grösster gemeinsamer Teiler ggT)
lcm	**l**east **c**ommon **m**ultiple (kleinstes gemeinsames Vielfaches kgV)
RSA	Asymmetrisches Kryptosystem von Ronald L. **R**ivest, Adi **S**hamir und Leonard **A**dleman

8 Quellen

[BGW88] Michael Ben-Or, Shafi Goldwasser und Avi Widgerson: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Proceedings of the 20th Annual ACMSymposium on Theory of Computing, 1988. ACM, New York, S. 1–10.

[BoFr01] Dan Boneh, Matthew Franklin: Efficient generation of shared RSA keys. EUROCRYPT 2001.

[NiSa11] Takashi Nishide & Kouichi Sakurai, Distributed Paillier Cryptosystem without Trusted Dealer. Springer-Verlag Berlin Heidelberg, 2011

[PaKr99] Pascal Paillier: Public-key cryptosystems based on composite degree residuosity classes. In EUROCRYPT 1999. LNCS, Vol. 1592, S. 223–238. Springer, Heidelberg (1999)

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In Advances in Cryptology - proceedings of EUROCRYPT ‘91, Lecture Notes in Computer Science, S. 522 – 526, 1991.

[WiLa11] Lagrange polynomial, Definition. Wikipedia [http://en.wikipedia.org/w/index.php?title=Lagrange_polynomial&oldid=426467984#Definition], 28.04.11

[WiPa11] Paillier cryptosystem, Wikipedia [http://en.wikipedia.org/w/index.php?title=Paillier_cryptosystem&direction=prev&oldid=420478569], 17.01.2011