Secret Sharing and Threshold Decryption

with the Paillier Cryptosystem

Einleitung

Das Ziel dieses Berichtes ist es, dem Leser welcher Grundkenntnisse von Kryptografie besitzt das Thema **Secret Sharing** und **Threshold Decryption with the Paillier Cryptosystem** näher zu bringen. Es wurde sehr viel Wert darauf gelegt, dass der Text verständlich ist. Dies wird vor allem mit ausführlichen Beispielen und Übungen erreicht.

> Wo wird Threshold Decryption im Bereich eVoting überhaupt eingesetzt?
>
> Am Besten kann man dies an Hand der Grafik aus *Advances in Cryptographic Voting Systems* von Ben Adida [BAVS06] zeigen:

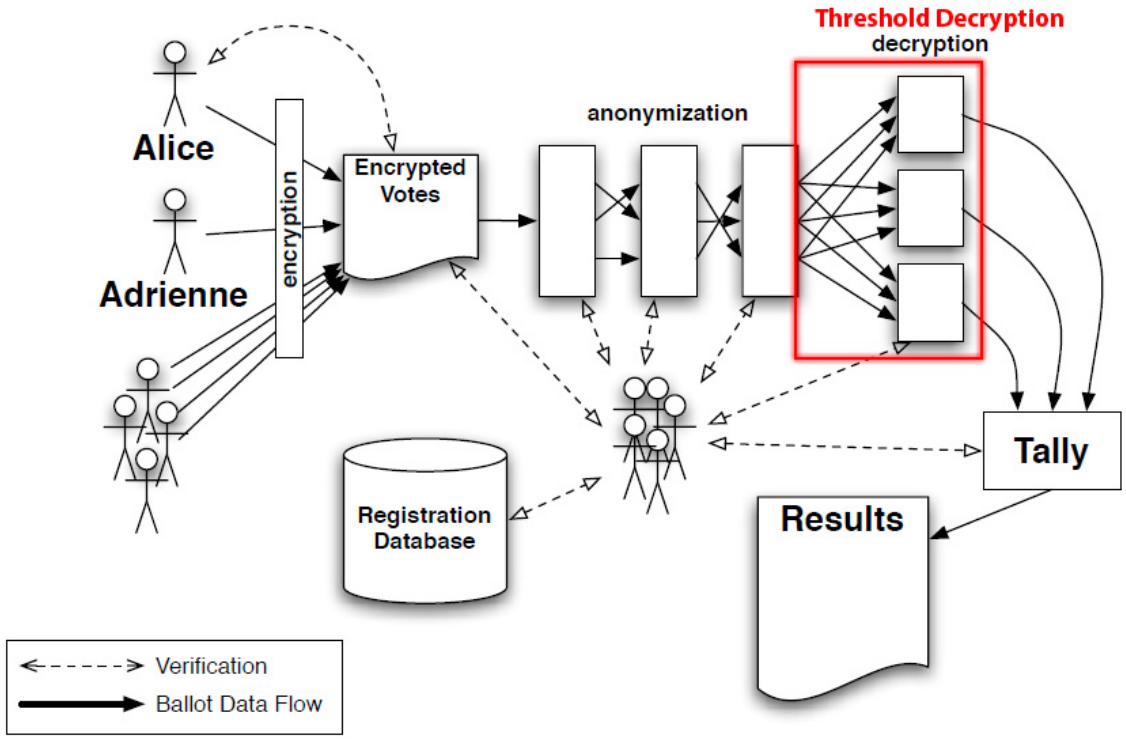

Abbildung 1: Threshold Decryption

Bevor jedoch auf das Thema Threshold Decryption mit Paillier eingegangen wird, wird zuerst das Prinzip von Secret Sharing erläutert, damit der Leser das Grundprinzip von Threshold Decryption versteht.

Secret Sharing

Bei einem klassischen Kryptosystem werden entweder ein Schlüssel, symmetrisches Kryptosystem, oder zwei Schlüssel, asymmetrisches Kryptosystem, verwendet. Dieses Verfahren eignet sich jedoch schlecht für eVoting: Wer soll den Schlüssel besitzen um die verschlüsselten Abstimmungszettel zu entschlüsseln? Es wäre schön wenn jede Partei seinen eigenen Schlüssel besässe und somit alle oder eine im Voraus bestimmte Mindestanzahl von Parteien die Abstimmungszettel entschlüsseln könnten. Man braucht einen Schlüssel welchen man teilen kann. Genau diese Technik des Aufteilens des Schlüssels wird **Secret Sharing** genannt.

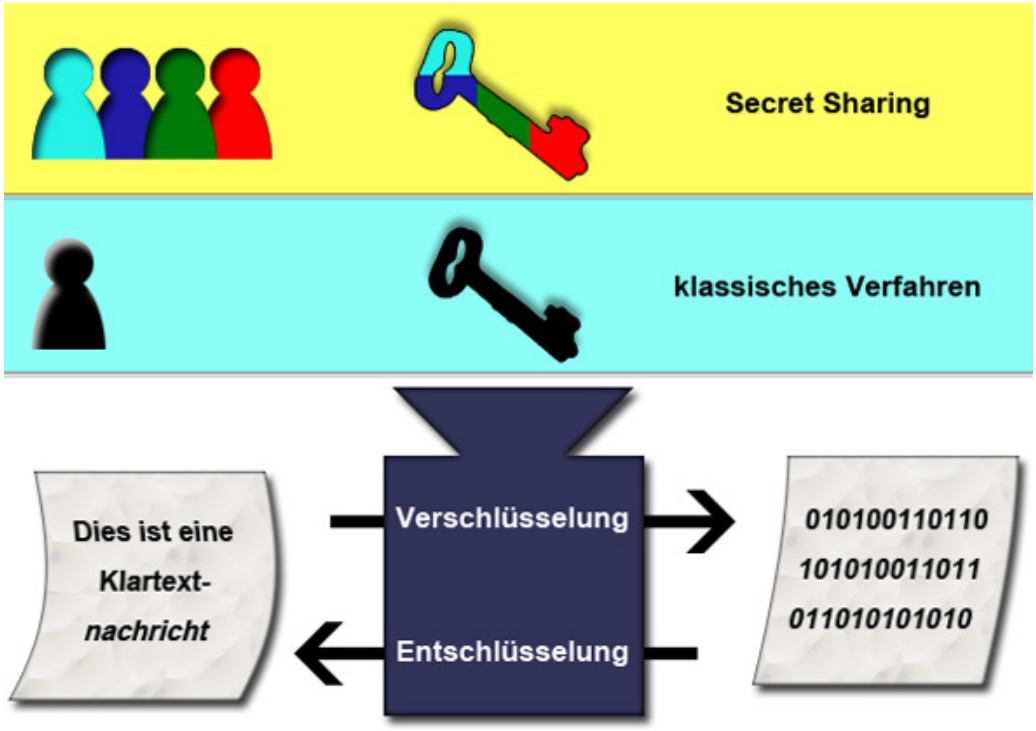

Abbildung 2: Secret Sharing

Damit man sich dies besser vorstellen kann wird Secret Sharing an Hand eines Zahlenbeispieles demonstriert.

Beispiel 1:
Secret Share durch Addition der einzelnen Schlüssel:

```
Schlüssel von Alice     s1 = 4
Schlüssel von Bob       s2 = 5
Schlüssel von Conny     s3 = 8
Schlüssel von Doug      s4 = 2
-----------------------------
Secret Share            s = 19
=============================
```

Das Problem dieses Verfahrens ist jedoch, dass die Bedingung, dass auch eine Mindestanzahl von Parteien das Geheimnis entschlüsseln kann, nicht erfüllt ist. Weigert sich eine Partei ihren Schlüssel zur Verfügung zu stellen, kann das Geheimnis nicht entschlüsselt werden. Dieses Verfahren ist ein **(n,n)-Schwellenwert-Schema**. Es sind *n* Parteien an diesem Verfahren beteiligt und es werden *n* Teilgeheimnisse benötigt um das Geheimnis rekonstruieren zu können.

Aufgabe 1:
Was wäre ein weiteres solches (n,n)-Schwellenwert-Schema?[1]

1 Lösungen zu den Aufgaben befinden sich am Ende des Berichtes.

Threshold Decryption

Was für das eVoting System benötigt wird ist ein (t,n)-Schwellenwert-Schema. Wobei *n* wieder die Anzahl beteiligter Parteien ist und *t* die Mindestanzahl Parteien die benötigt werden um das Geheimnis zu entschlüsseln. Ein solches Verfahren wird **Threshold Decryption** genannt. Wie Threshold Decryption funktioniert kann man anschaulich am Prinzip von **Shamir's Secret Sharing** aufzeigen.
Bei Shamir braucht man einen Dealer, diesem Dealer müssen alle Parteien vertrauen, da er die einzelnen Teilgeheimnisse kennt und verteilt. Der Dealer bestimmt ein Polynom vom Grad *t*-1:

$$f(x) = s + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$$

Wobei *s* dem Geheimnis entspricht. Die a_i können vom Dealer beliebig gewählt werden. Der Dealer kann nun eine beliebige Anzahl Wertepaare $(x, f(x))$ *n* erstellen. Wobei x_i öffentlich sein darf. $f(x)$ hingegen ist ein Share und muss geheim gehalten werden, es wird auch als s_i bezeichnet.
Nach dem Fundamentalsatz der Algebra braucht man mindestens *t* Wertepaare um ein Polynom, und somit auch *s*, eindeutig zu bestimmen.

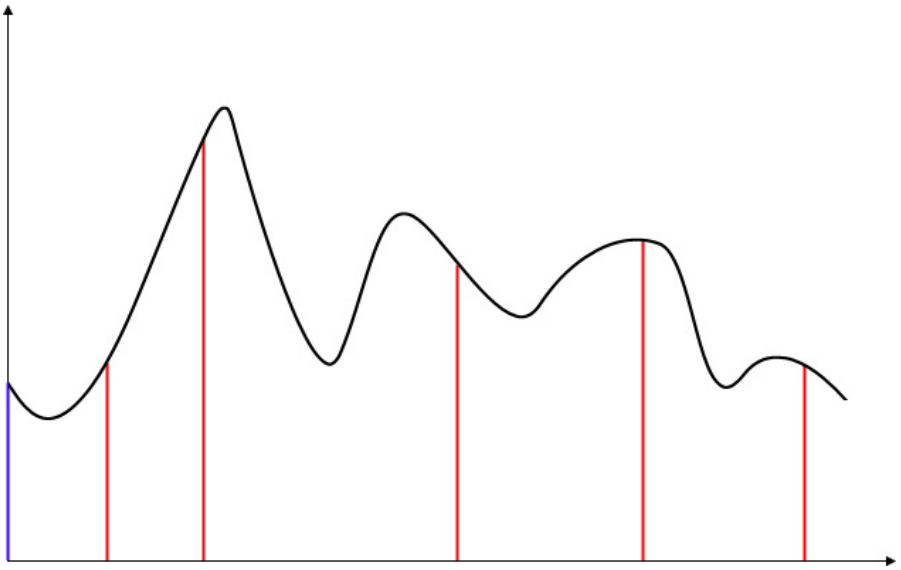

Abbildung 3: Shamir

Diese Grafik zeigt ein Polynom bei welchem der y-Wert bei $x = 0$ blau markiert wurde und bei Shamir dem Geheimnis *s* entspricht. Die roten Säulen sind Stützwerte welche an die einzelnen Parteien durch den Dealer verteilt werden. Bei diesen Stützsäulen sind die x-Werte öffentlich, die y-Werte zu den entsprechenden x-Werten sind jedoch streng geheim zu halten und nur der entsprechenden Partei, und in dieser Variante leider auch dem Dealer, bekannt. Hat man genügend (x, y)-Paare kann man das Polynom lösen und somit auch das Geheimnis *s* bestimmen.

Das Geheimnis *s* kann man einfach und präzise mit Hilfe von Lagrange berechnen, welches am folgenden Beispiel demonstriert wird.

Beispiel 2:
Es sind Bundesratswahlen und wir wollen das Resultat mit mindestens vier Teilgeheimnissen von vier Parteien berechnen können, *t* entspricht 4. Im Gesamten sind es sieben Parteien, *n* entspricht 7. Der Dealer, welchem alle Parteien vertrauen, z.B.: Silvio Berlusconi erstellt ein Polynom dritten Grades, da *t*-1 = 3 ist. Er erstellt folgendes Polynom:

$$f(x) = 4 + 12x + 6x^2 + 5x^3$$

In diesem Fall ist das Geheimnis *s* = 4.

Silvio verteilt nun folgende Wertepaare an die Parteien:

SVP	(1, 27)
FDP	(2, 92)
CVP	(3,229)
SP	(4, 468)
SD	(5, 839)
Grüne	(6, 1372)
EVP	(7, 2097)

Die kleineren Parteien vermuten, dass sie wieder keinen Bundesratsitz bekommen und weigern sich ihr Teilgeheimnis zu berechnen und zu veröffentlichen. Die grossen vier Parteien sind jedoch im Stande das Geheimnis alleine zu berechnen, da wir definiert haben, dass lediglich vier Parteien benötigt werden um das Geheimnis zu entschlüsseln. Die Parteien tauschen ihr öffentliches x untereinander aus und jede Teilpartei berechnet unabhängig der anderen Parteien ihr Teilgeheimnis wie folgt:

$$g(\mathrm{x}) = \mathrm{s}_i \cdot \prod_{i \neq j} \frac{\mathrm{x} - \mathrm{x}_j}{\mathrm{x}_i - \mathrm{x}_j}$$

Wobei $g(x)$ dem Wert an der Stelle a_x entspricht. Da aber nur das Geheimnis *s* von Interesse ist, muss nur $g(0)$ berechnet werden. Jede Partei berechnet mit ihrem eigenen Share s_i das Teilgeheimnis:

$$g(0) = \mathrm{s}_i \cdot \prod_{i \neq j} \frac{0 - \mathrm{x}_j}{\mathrm{x}_i - \mathrm{x}_j} = \mathrm{s}_i \cdot \prod_{i \neq j} \frac{\mathrm{x}_j}{\mathrm{x}_j - \mathrm{x}_i}$$

$$\mathrm{SVP} = 27 \cdot \prod_{i \neq j} \frac{\mathrm{x}_j}{\mathrm{x}_j - \mathrm{x}_i} = 27 \cdot \frac{2}{2-1} \cdot \frac{3}{3-1} \cdot \frac{4}{4-1} = \underline{108}$$

$$\mathrm{FDP} = 92 \cdot \prod_{i \neq j} \frac{\mathrm{x}_j}{\mathrm{x}_j - \mathrm{x}_i} = 92 \cdot \frac{1}{1-2} \cdot \frac{3}{3-2} \cdot \frac{4}{4-2} = \underline{-552}$$

$$\mathrm{CVP} = 229 \cdot \prod_{i \neq j} \frac{\mathrm{x}_j}{\mathrm{x}_j - \mathrm{x}_i} = 229 \cdot \frac{1}{1-3} \cdot \frac{2}{2-3} \cdot \frac{4}{4-3} = \underline{916}$$

$$\mathrm{SP} = 468 \cdot \prod_{i \neq j} \frac{\mathrm{x}_j}{\mathrm{x}_j - \mathrm{x}_i} = 468 \cdot \frac{1}{1-4} \cdot \frac{2}{2-4} \cdot \frac{3}{3-4} = \underline{-468}$$

Wichtig hierbei ist, dass nur die x-Werte der Parteien berücksichtigt werden die auch bereit sind ihr Teilgeheimnis zu berechnen. Die Parteien können nun zusammen das Geheimnis *s* berechnen:

$$\mathrm{s} = 108 - 552 + 916 - 468 = \underline{4}$$

Dieses kann Silvio, der Dealer, bestätigen.

Das Problem bei diesem Verfahren ist, dass man einen Dealer braucht dem man vertrauen muss. Es wäre von Vorteil, wenn jede Partei ihr Teilgeheimnis selbständig und unabhängig eines Dealers wählen könnte.

Konkrete Threshold Decryption kann mit Hilfe folgender bekannter Kryptosysteme erstellt werden:

- RSA
- Paillier
- Damgård-Jurik
- El-Gamal

Aufgabe 2:
Berechne die Teilgeheimnisse s_i aus der oben aufgeführten Aufgabe, wenn die EVP, SD, CVP, FDP und die Grünen bereit sind ihr Teilgeheimnis zu veröffentlichen.

Threshold Decryption with the Paillier Cryptosystem

Auf das Paillier Cryptosystem im Speziellen wird in diesem Bericht nicht mehr eingegangen, da dies Bestandteil eines anderen Berichtes ist. Dieses Kapitel basiert auf *A Generalization of Paillier's Public-Key System with Applications to Electronic Voting* [DaJuNi].

Der Fokus liegt bei der Threshold Decryption mit Hilfe des Paillier Cryptosystems. Daher wurde dieser Teil in folgende vier Unterkapitel eingeteilt:

- **Schlüsselerzeugung**
  Dieser Bereich behandelt die Erstellung von l Teilschlüsseln wovon mindestens ω Teilnehmer ihr mit Hilfe ihres persönlichen Schlüssels berechnetes Teilgeheimnis zur Verfügung stellen müssen.
- **Verschlüsselung**
  Dieser Bereich behandelt, wie ein Cipher c aus einer Nachricht M erstellt wird.
- **Teilgeheimnis Entschlüsselung**
  Wie die einzelnen Teilnehmer ihr Teilgeheimnis aus dem Cipher c bestimmen können wird in diesem Bereich behandelt.
- **Gesamtentschlüsselung**
  Wie danach aus mindestens ω Teilgeheimnissen das Gesamtgeheimnis berechnet werden kann deckt dieses Unterkapitel ab.

Schlüsselerzeugung

Das generalisierte Paillier Verfahren [DaJuNi] unterscheidet sich bei der Schlüsselerzeugung insofern vom klassischen Paillier Verfahren, dass nicht zwei klassische Primzahlen sondern zwei sogenannte sichere Primzahlen (safe primes) verwendet werden. Sichere Primzahlen sind nicht zu verwechseln mit starken Primzahlen (strong primes) welche man in der Kryptografie ziemlich oft benötigt. Das Eine ist auch keine Untermenge des Anderen. Dies sieht man schnell wenn man die ersten vier starken Primzahlen mit den ersten vier sicheren Primzahlen vergleicht:

Strong primes: 11, 17, 29, 37
Safe primes: 5, 7, 11, 23

Da für diese Schlüsselerzeugung lediglich sichere Primzahlen benötigt werden, wird nicht weiter auf die starken Primzahlen eingegangen. Eine sichere Primzahl muss folgende Bedingung erfüllen:

$$p' = \frac{p-1}{2}$$

Wobei p einer sicheren Primzahl entspricht und p' einer klassischen Primzahl.

Für die Schlüsselerzeugung mit dem generalisierten Paillier Verfahren benötigen wir zwei sichere Primzahlen mit ihren entsprechenden klassischen Primzahlen. Die einfachste Art diese zu erhalten ist es zuerst die sicheren Primzahlen zu bestimmen und danach zu überprüfen ob die entsprechenden p's auch wirklich Primzahlen sind.

Beispiel 3:
Bestimmung zweier sicherer Primzahlen mit ihren entsprechenden klassischen Primzahlen.
Als erstes wird die Bedingung der sicheren Primzahlen umgeformt, damit von den sicheren Primzahlen ausgegangen werden kann.

$$\mathrm{p}' = \frac{\mathrm{p}-1}{2} \rightarrow \mathrm{p} = \underline{2\mathrm{p}' + 1}$$

Nun wird zufällig eine Primzahl p' gewählt und geprüft ob $2p' + 1$ einer Primzahl entspricht.

$\mathrm{p}' = 17 \rightarrow 17 \cdot 2 + 1 = \underline{35}$ (keine Primzahl)
$\mathrm{p}' = 19 \rightarrow 19 \cdot 2 + 1 = \underline{39}$ (keine Primzahl)
$\mathbf{p' = 23 \rightarrow 23 \cdot 2 + 1 = \underline{47}}$ **(ist eine Primzahl)**

Das erste Paar ist also $p' = 23; p = 47$

$\mathbf{q' = 29 \rightarrow 29 \cdot 2 + 1 = \underline{59}}$ **(ist eine Primzahl)**

Das zweite Paar ist $q' = 29; q = 59$

Mit Hilfe dieser Primzahlen ist es nun möglich die beiden Parameter n und m zu bestimmen welche für den weiteren Verlauf der Schlüsselerzeugung benötigt werden.

$n = p \cdot q = 47 \cdot 59 = \underline{2'773}$
$m = p' \cdot q' = 23 \cdot 29 = \underline{667}$

Mit Hilfe eines Exponenten s kann die Grösse des Klartextes anpasst werden. Die Grösse des Klartextes entspricht dabei Z_{n^s}. Die Bezeichnung s könnte hier etwas verwirrend sein, da dieses s nicht geheim (secret) ist, es ist allen Parteien bekannt.

Der geheime Exponent d muss für die Threshold Decryption zwei Bedingungen erfüllen:

$d = 0 \; mod \; m$
$d = 1 \; mod \; n^s$

Dieser kann mit einer Kombination aus dem Erweiterten Euklidischen Algorithmus und dem Chinesischen Restsatz berechnet werden.

Beispiel 4:
Hier wird das Beispiel 3 weitergeführt, es wird nun d berechnet. Da angenommen wird, dass nur eine sehr kleine Nachricht verschickt wird, wird $s = 1$ gewählt.

Der Chinesische Restatz liefert:

$d = 0 \; mod \; m = 0 \; mod \; 667$
$d = 1 \; mod \; n^s = 1 \; mod \; 2'773$
$M = m_1 \cdot m_2 = 667 \cdot 2'773 = \underline{1'849'591}$
$M_1 = 1'849'591 \div 2'773 = \underline{667}$
$M_2 = 1'849'591 \div 667 = \underline{2'773}$
$2'773 \cdot x + 667 \cdot y = 1$

Mit Hilfe des Erweiterten Euklidischen Algorithmus kann diese Gleichung wie folgt gelöst werden:

$2'773 = 4 \cdot 667 + 105$
$667 = 6 \cdot 105 + 37$
$105 = 2 \cdot 37 + 31$
$37 = 1 \cdot 31 + 6$
$31 = 5 \cdot 6 + 1$
$1 = 31 - 5 \cdot 6$
$= 31 - 5 \cdot (37 - 1 \cdot 31) = 6 \cdot 31 - 5 \cdot 37$
$= 6 \cdot (105 - 2 \cdot 37) - 5 \cdot 37 = 6 \cdot 105 - 17 \cdot 37$
$= 6 \cdot 105 - 17 \cdot (667 - 6 \cdot 105) = 108 \cdot 105 - 17 \cdot 667$
$= 108 \cdot (2773 - 4 \cdot 667) - 17 \cdot 667 = \underline{108} \cdot 2773 \underline{-449} \cdot 667$

Diese Werte können nun im Chinesischen Restsatz verwendet werden:

$e_1 = 667 \cdot -449 = \underline{-299'483}$
$e_2 = 2773 \cdot 108 = \underline{299'484}$

Aus diesen Daten folgt für d:

$d = 1 \cdot -299'483 + 0 \cdot 299'484 = \underline{-299'483}$
$-299'483 \; mod \; 1'849'591 = \underline{1'550'108}$

Dieses Resultat wird nun auf die oben gestellten Bedingungen getestet:

$1'550'108 \; mod \; 2773 = 1$ √ **erfüllt**
$1'550'108 \; mod \; 667 = 0$ √ **erfüllt**

Was auffällt ist, dass nicht immer die gesamte Berechnung durchgeführt werden müsste. Der Erweiterte Euklidische Algorithmus würde vollkommen ausreichen. Man muss aber wissen, welches Resultat das Richtige ist. In diesem Fall beispielsweise liefert der Algorithmus:

$$e_1 = 667 \cdot -449 = -299'483 = \underline{1'550'108}$$
$$e_2 = 2773 \cdot 108 = \underline{299'484}$$

Da $1'550'108 \neq 299'484$ sehen wir, dass es darauf ankommt.
Durch Umformung dieser Berechnungen erhält man:
$d = m \cdot x \; mod \; m \cdot n^s$, wobei x dem Resultat aus $m \cdot x + n^s \cdot y = 1$ entspricht.

Wenn d bestimmt wurde, kann mit Hilfe von Shamir ein Polynom bestimmt werden. Dieses Polynom besitzt die Form:

$$f(x) = \sum_{i=0}^{\omega-1} a_i \cdot x^i \bmod m \cdot n^s$$

In diesem Fall entspricht das Geheimnis a_0 dem geheimen Exponenten d.

Beispiel 5:

l	Anzahl Teilschlüssel	8
ω	Mindestanzahl Teilnehmer	5

Es werden zufällige Werte für die a_i bestimmt:

$$f(x) = 1'550'108 + 12x + 5x^2 + 84x^3 + 115x^4$$

An Hand dieses Polynoms können nun l Wertepaare erstellt werden:

s_1: (1, 1‘550‘324)
s_2: (2, 1‘552‘664)
s_3: (3, 1‘561‘772)
s_4: (4, 1‘585‘052)
s_5: (5, 1‘632‘668)
s_6: (6, 1‘717‘544)
s_7: (7, 1‘855‘364)
s_8: (8, 2‘064‘572)

Verschlüsselung

Der verschlüsselte Text c wird wie folgt aus der Nachricht M generiert:

$$c = (n+1)^M \cdot r^{n^s} \; mod \; n^{s+1}$$

Wobei r einer zufälligen Zahl aus Z_n^* entspricht. Diese Variable wird benötigt, damit aus gleichen Klartexten unterschiedliche Ciphertexte entstehen.

Beispiel 6:
Hier soll ein Klartext 1337 mit Hilfe des generalisierten Paillier-Verfahren verschlüsselt werden. Zusätzlich wird ein r bestimmt, welches für die Verschlüsselung benötigt wird.

M = 1337
r = 3
$$c = (2'773 + 1)^{1337} \cdot 3^{2773^1} \; mod \; 2'773^{1+1}$$
$$c = 2'774^{1337} \cdot 3^{2773} \; mod \; 2'773^2$$

Hohe Potenzen berechnet man am besten, wenn man die Potenzen aufteilt und diese dann multipliziert. Der schnellste Weg ist es, den Exponenten in die Binärdarstellung zu überführen. Ein sehr einfacher Algorithmus für diese Transformation ist es die Dezimalzahl durch 2 zu dividieren und falls sich die Zahl dividieren lässt sich eine 0 zu notieren, ansonsten, bei einer Division mit Rest, wird der Rest mit 2 multipliziert. Dieser Rest ergibt immer 1, da es nur einen Rest von 0.5 ergeben kann.

Danach muss man diese Zahlen nur noch in umgekehrter Reihenfolge notieren und man erhält die gesuchte Binärdarstellung:

$1337 \div 2 = 668.5$	→ 1	*1*
$668 \div 2 = 334$	→ 0	*2*
$334 \div 2 = 167$	→ 0	*4*
$167 \div 2 = 83.5$	→ 1	*8*
$83 \div 2 = 41.5$	→ 1	*16*
$41 \div 2 = 20.5$	→ 1	*32*
$20 \div 2 = 10$	→ 0	*64*
$10 \div 2 = 5$	→ 0	*125*
$5 \div 2 = 2.5$	→ 1	*256*
$2 \div 2 = 1$	→ 0	*512*
	→ 1	*1024*

1337 = <u>10100111001</u>

Anhand dieser Binärdarstellung kann man den Exponenten folgendermassen aufteilen:

$$2'774^{1337} = 2'774^{1} \cdot 2'774^{8} \cdot 2'774^{16} \cdot 2'774^{32} \cdot 2'774^{256} \cdot 2'774^{1024}$$

Nun werden die Quadrate bis „$berechnete\ Exponent \cdot 2 = grösste\ gesuchte\ Exponent$“ berechnet:

$$2'774^{1}\ mod\ 2773^{2} = 2'774\ mod\ 7'689'529$$
$$2'774^{2}\ mod\ 2773^{2} = 2'774^{1} \cdot 2'774^{1} = 5'547\ mod\ 7'689'529$$
$$2'774^{4}\ mod\ 2773^{2} = 2'774^{2} \cdot 2'774^{2} = 5'547 \cdot 5'547\ mod\ 7'689'529$$
$$= 11'093\ mod\ 7'689'529$$
$$2'774^{8}\ mod\ 2773^{2} = 2'774^{4} \cdot 2'774^{4} = 22'185\ mod\ 7'689'529$$
$$2'774^{16}\ mod\ 2773^{2} = 2'774^{8} \cdot 2'774^{8} = 44'369\ mod\ 7'689'529$$
$$2'774^{32}\ mod\ 2773^{2} = 2'774^{16} \cdot 2'774^{16} = 88'737\ mod\ 7'689'529$$
$$2'774^{64}\ mod\ 2773^{2} = 2'774^{32} \cdot 2'774^{32} = 177'473\ mod\ 7'689'529$$
$$2'774^{128}\ mod\ 2773^{2} = 2'774^{64} \cdot 2'774^{64} = 354'945\ mod\ 7'689'529$$
$$2'774^{256}\ mod\ 2773^{2} = 2'774^{128} \cdot 2'774^{128} = 709'889\ mod\ 7'689'529$$
$$2'774^{512}\ mod\ 2773^{2} = 2'774^{256} \cdot 2'774^{256} = 1'419'777\ mod\ 7'689'529$$
$$2'774^{1024}\ mod\ 2773^{2} = 2'774^{512} \cdot 2'774^{512} = 2'839'553\ mod\ 7'689'529$$

Diese Werte werden nun in die ursprüngliche Gleichung eingesetzt:

$$2'774 \cdot 22'185 \cdot 44'369 \cdot 88'737 \cdot 709'889 \cdot 2'839'553\ mod\ 7'689'529 = \underline{3'707'502}$$

<u>Aufgabe 3</u>
Berechnen Sie $3^{2773}\ mod\ 2773^{2}$.

Diese Resultate werden nun in die Gleichung für den Ciphertext *c* eingesetzt:

$$c = 2'774^{1337} \cdot 3^{2773}\ mod\ 2773^{2} = 3'707'502 \cdot 1'683'858\ mod\ 2773^{2} = \underline{1'303'957}$$

Teilgeheimnis Entschlüsselung

Das Teilgeheimnis c_i wird wie folgt berechnet:

$$c_i = c^{2 \cdot \Delta s_i} \; mod \; n^{s+1}$$

Wobei $\Delta = l!$ entspricht.

Beispiel 7
Es sollen die 8 Teilgeheimnisse berechnet werden:

s_1: (1, 1‘550‘324)
s_2: (2, 1‘552‘664)
s_3: (3, 1‘561‘772)
s_4: (4, 1‘585‘052)
s_5: (5, 1‘632‘668)
s_6: (6, 1‘717‘544)
s_7: (7, 1‘855‘364)
s_8: (8, 2‘064‘572)

$$c_1 = 1'303'957^{2 \cdot 40320 \cdot 1550324} \; mod \; 2'773^2 = \underline{5'688'632}$$
$$c_2 = 1'303'957^{2 \cdot 40320 \cdot 1552664} \; mod \; 2'773^2 = \underline{4'538'451}$$
$$c_3 = 1'303'957^{2 \cdot 40320 \cdot 1561772} \; mod \; 2'773^2 = \underline{2'472'942}$$
$$c_4 = 1'303'957^{2 \cdot 40320 \cdot 1585052} \; mod \; 2'773^2 = \underline{311'067}$$
$$c_5 = 1'303'957^{2 \cdot 40320 \cdot 1632668} \; mod \; 2'773^2 = \underline{7'596'501}$$
$$c_6 = 1'303'957^{2 \cdot 40320 \cdot 1717544} \; mod \; 2'773^2 = \underline{1'902'329}$$
$$c_7 = 1'303'957^{2 \cdot 40320 \cdot 1855364} \; mod \; 2'773^2 = \underline{1'391'060}$$
$$c_8 = 1'303'957^{2 \cdot 40320 \cdot 2064572} \; mod \; 2'773^2 = \underline{1'948'292}$$

Gesamtentschlüsselung

Bei der Gesamtentschlüsselung müssen mindestens ω Parteien ihr Geheimnis preisgeben, damit das Gesamtgeheimnis entschlüsselt werden kann. Diese Entschlüsselung wird wie folgt durchgeführt.

$$c' = \prod_{i \in S} c_i^{2 \cdot \lambda_{0,i}^S} \; mod \; n^{s+1} \qquad wobei \;\; \lambda_{0,i}^S = \Delta \prod_{j \in S \setminus i} \frac{j}{j-i}$$

Der hintere Teil dieser Berechnung, $\prod_{j \in S \setminus i} \frac{j}{j-i}$, sollte noch präsent sein, hier wird das Prinzip von Shamir für die Threshold Decryption mit dem Paillier Verfahren angewendet.

Beispiel 8
Um $c‘$ zu berechnen werden für diese Berechnung die Zahlen aus den vorherigen Beispielen eingesetzt. Womit wir folgendes Resultat erhalten:

$$c' = \underline{6‘169‘926}$$

Mit Hilfe von c' können wir den Klartext rekonstruieren.

$$M = c' \cdot (4 \cdot \Delta^2)^{-1} \; mod \; n^s = \underline{1337}$$
$$4 \cdot \Delta^2 = 4 \cdot 40'320^2 = \underline{6'502'809'600}$$

Das modulare Inverse wird mit Hilfe des Erweiterten Euklidischen Algorithmus berechnet.

$$x \cdot 6'502'809'600 \; + \; y \cdot 27'732 = \underline{164‘986}$$

Dieses Kapitel hat somit den gesamten Ablauf des generalisierten Paillier Verfahren [DaJuNi] erläutert und an Hand von Beispiel-Daten demonstriert. Das Problem dieses Verfahrens ist immer noch, dass ein Dealer, welche zusätzlich zu den einzelnen Parteien alle benötigten Informationen besitzt, existiert. Das nächste Kapitel wird auf dieses Problem eingehen.

Verteilte Schlüssel ohne Dealer

Dieses Kapitel behandelt die Erstellung eines robusten, effizienten und sicheren verteilten Schlüssels. Dies ermöglicht auf den Dealer zu verzichten. Es wird Bezug auf *Robust Efficient Distributed RSA-Key Generation* [FrMKYu] genommen. Diese Lösung setzt voraus, dass $n \geq 2t + 1$ ist, wobei bei maximal t Parteien Probleme auftauchen dürfen. Desweiteren werden folgende Techniken benötigt:

- Ein Multiplikations-Protokoll für eines dieser beiden Probleme.
  - Ein Problem welches auf dem *discrete logarithm* Problem basiert.
  - Ein Problem welches auf dem RSA Problem basiert.
- Einen Mechanismus der garantiert, dass Informationen welche man teilen muss über die gesamte Zeit hinweg konsistent bleiben, sogar wenn diese Informationen verändert werden.
- Sowie einen *commitment* Mechanismus.

Es werden bestimmte Parameter auftauchen welche hier kurz erläutert werden:

h	Der Sicherheitsparamter
H	2^h
N_*	Ist das Produkt zweier unbekannter Primzahlen im Bereich $[\sqrt{H}, 2\sqrt{H}]$
g_*, h_*	Sind Generatoren bei welchen der discrete log $mod\ N_*$ unbekannt ist

Das grosse Problem hierbei ist das Finden eines solchen N_*. Hat man ein solches N_* gefunden hat man auch das Problem dieses Kapitels gelöst. Auf Grund dieser einführenden Informationen wählt sich jede Partei ein Triple (N_k, g_k, h_k) und veröffentlicht es.

Um eine grosse Primzahl zu erstellen benötigen wir die Arbeit von Boneh and Franklin, welche besagt, dass;

$$(\emptyset(N))^{-1}\ mod\ e$$

Wobei e eine starke Primzahl (strong prime) ist.

$$E = 2e + 1$$

Der Ablauf sieht nun wie folgt aus:

1. Die Parteien wählen ein e zufällig und testen ob $2e + 1$ auch eine Primzahl ist.
2. Die Parteien wählen ein g_e, $h_e \in_R Z_E^*$.
3. Die Parteien wählen $g, h \in Z_N^*$.
4. Die Parteien wählen $m_i \in_R [0, H]$ und führen Shamir von m_i aus.

   $$m = m_1 + \cdots + m_n$$

5. Das Multiplikations-Protokoll berechnet nun $D = L^4\emptyset(N)m\ mod\ e$ mit allen Werten von Shamir. Dabei kann jedes $m_i\ mod\ e$ genommen werden.

   Für $1 \leq i \leq n$, $f_i(x) = a_i(x) + b_i(x) = \sum_{j=0}^{t} f_{i,j}x^j$ wobei $a_i(x)$ und $b_i(x)$ benötigt werden um p_i und q_i zu berechnen. Diese wiederum werden benötigt um einen Algorithmus h-Mal durchzuführen bis alle Parteien bestätigen können, dass N ein Produkt von zwei Primzahlen ist. Dieser Algorithmus wird genauer in [FrMKYu] unter Punkt *8 Robust Distributed Double-Primality Test* beschrieben.

   $$\emptyset(N) = N - \sum_{i=1}^{n}(p_i + q_i) + 1$$

   Wir nehmen nun $f(x) = L^2(N + 1) - \sum_{i=1}^{n} f_i(x)$ daraus folgt $f(0) = L^2\emptyset(N)$

6. Jede Partei i multipliziert nun ihre Geheimnisse m_j mit $D^{-1}\ mod\ e$, dieses enthält ein Geheimnis $W \equiv (L^4\emptyset(N))^{-1}\ mod\ e$.

7. Nun wird eine Multiplikation über N ausgeführt um ein Polynom $-L^4W\emptyset(N)+1$ zu erhalten.

$$v(0) \equiv 0\ mod\ e$$
$$v(0) \equiv 1 mod\ \emptyset(N)$$

8. Die Parteien führen den Multiplikations-Algorithmus für $t+1$ Parteien durch.
9. Diese $t+1$ Parteien teilen ihre Geheimnisse mit e und veröffentlichen ihren Rest. Die Summe dieser Reste wird zu einem Vielfachen von e. Nun muss eine beliebige Partei nur noch diese Reste zu ihrem Geheimnis addieren. Wendet man nun den Multiplikations-Algorithmus auf dieses Resultat an, erhält man ein (t,n)-Polynom für die Verteilung von d.

Mit Hilfe dieses Verfahren ist es möglich alle zu Beginn definierten Anforderungen an die Threshold Decryption zu erfüllen.

Lösungen

Aufgabe 1:
Wenn man an Stelle der Addition eine XOR-Operation durchführt.

Aufgabe 2:

$$EVP = 2097 \cdot \prod_{i \neq j} \frac{x_j}{x_j - x_i} = 2097 \cdot \frac{5}{5-7} \cdot \frac{3}{3-7} \cdot \frac{2}{2-7} \cdot \frac{6}{6-7} = \underline{\mathbf{18'873/2}}$$

$$SD = 839 \cdot \prod_{i \neq j} \frac{x_j}{x_j - x_i} = 839 \cdot \frac{7}{7-5} \cdot \frac{3}{3-5} \cdot \frac{2}{2-5} \cdot \frac{6}{6-5} = \underline{\mathbf{17'619}}$$

$$CVP = 229 \cdot \prod_{i \neq j} \frac{x_j}{x_j - x_i} = 229 \cdot \frac{7}{7-3} \cdot \frac{5}{5-3} \cdot \frac{2}{2-3} \cdot \frac{6}{6-3} = -\underline{\mathbf{8'015/2}}$$

$$FDP = 92 \cdot \prod_{i \neq j} \frac{x_j}{x_j - x_i} = 92 \cdot \frac{7}{7-2} \cdot \frac{5}{5-2} \cdot \frac{3}{3-2} \cdot \frac{6}{6-2} = \underline{\mathbf{966}}$$

$$Grüne = 1372 \cdot \prod_{i \neq j} \frac{x_j}{x_j - x_i} = 1372 \cdot \frac{7}{7-6} \cdot \frac{5}{5-6} \cdot \frac{3}{3-6} \cdot \frac{2}{2-6} = \underline{\mathbf{-24'010}}$$

Kontrolle: $\frac{18'873}{2} + 17'619 - \frac{8'015}{2} + 966 - 24'010 = \underline{4}$

Aufgabe 3

$3^{2773}\ mod\ 2773^2$

Binärdarstellung des Exponenten:

$2773 \div 2 = 1386.5$	→ 1	*1*
$1386 \div 2 = 693$	→ 0	*2*
$693 \div 2 = 346.5$	→ 1	*4*
$346 \div 2 = 173$	→ 0	*8*
$173 \div 2 = 86.5$	→ 1	*16*
$86 \div 2 = 43$	→ 0	*32*
$43 \div 2 = 21.5$	→ 1	*64*
$21 \div 2 = 10.5$	→ 1	*128*
$10 \div 2 = 5$	→ 0	*256*
$5 \div 2 = 2.5$	→ 1	*512*
$2 \div 2 = 1$	→ 0	*1024*
	→ 1	*2048*

2773=101011010101

$3^1 = 3\ mod\ 2773^2$
$3^2 = 3^1 \cdot 3^1 = 9\ mod\ 7'689'529$
$3^4 = 9 \cdot 9 = 81\ mod\ 7'689'529$
$3^8 = 81 \cdot 81 = 6'561\ mod\ 7'689'529$
$3^{16} = 6'561 \cdot 6'561 = 4'599'076\ mod\ 7'689'529$
$3^{32} = 4'599'076 \cdot 4'599'076 = 4'907'824\ mod\ 7'689'529$
$3^{64} = 4'907'824 \cdot 4'907'824 = 1'948'673\ mod\ 7'689'529$
$3^{128} = 1'948'673 \cdot 1'948'673 = 6'354'859\ mod\ 7'689'529$
$3^{256} = 6'354'859 \cdot 6'354'859 = 3'099'818\ mod\ 7'689'529$
$3^{512} = 3'099'818 \cdot 3'099'818 = 5'436'608\ mod\ 7'689'529$
$3^{1024} = 5'436'608 \cdot 5'436'608 = 2'556'624\ mod\ 7'689'529$
$3^{2048} = 2'556'624 \cdot 2'556'624 = 3'631'035\ mod\ 7'689'529$

$3^{2773}\ mod\ 2773^2 = 3^1 \cdot 3^4 \cdot 3^{16} \cdot 3^{64} \cdot 3^{128} \cdot 3^{512} \cdot 3^{2048}\ mod\ 2773^2$
$= 3 \cdot 81 \cdot 4'599'076 \cdot 1'948'673 \cdot 6'354'859 \cdot 5'436'608 \cdot 3'631'035\ mod\ 2773^2$
$= 1'683'858$

Referenzen

[BAVS06] *Advances in Cryptographic Voting Systems*, 31.08.2006
Ben Adida
http://groups.csail.mit.edu/cis/theses/adida-phd.pdf

[DaJuNi] *A Generalization of Paillier's Public-Key System with Applications to Electronic Voting*
Ivan Damgård, Mads Jurik und Jesper Buus Nielsen
http://www.daimi.au.dk/~ivan/GenPaillier_finaljour.ps

[FrMKYu] *Robust Efficient Distributed RSA-Key Generation*
Yair Frankel, Philip D. MacKenzie und Moti Yung
https://eprints.kfupm.edu.sa/62836/1/62836.pdf

[WiLa] *Wikipedia Lagrange*, 05.01.2010
http://de.wikipedia.org/wiki/Polynominterpolation

[WiSaP] *Wikipedia Safe Prime*, 05.01.2010
http://en.wikipedia.org/wiki/Safe_prime

[WiSh] *Wikipedia Shamir's Secret Sharing*, 05.01.2010
http://de.wikipedia.org/wiki/Shamirs_Secret_Sharing

[WiSS] *Wikipedia Secret Sharing*, 05.01.2010
http://de.wikipedia.org/wiki/Secret_Sharing

[WiStP] *Wikipedia Strong Prime*, 05.01.2010
http://en.wikipedia.org/wiki/Strong_prime