Universität Konstanz
Fachbereich Mathematik und Statistik
Prof. Dr. Salma Kuhlmann
Dr. Lothar Sebastian Krapp, Michele Serra

Sommersemester 2021

RSA-Verfahren

Autor: Sophia-Elena Mauch
Datum: 23.06.2021

Supervisor: Michele Serra

Abstract:

In diesem Vortrag zum *RSA-Verfahren* soll es um die Funktionsweise sowie die mathematischen Grundlagen dieses effizienten Verschlüsselungsverfahrens gehen. Zum besseren Verständnis des prinzipiellen Ablaufs liegt am Anfang dieser Abfassung der Fokus auf der Darstellung einer überblicksartigen und schematisch aufgearbeiteten Erklärung des Verfahrens. Hiernach werden die diesem Verfahren zugrundeliegenden mathematischen Gesetzmäßigkeiten herausgearbeitet – hierzu muss tiefer auf die Themengebiete *(große) Primzahlen und Primzahlfindung, Lösen modularer Potenzgleichungen* sowie den *Satz von Euler* eingegangen werden. Den Abschluss bildet ein kurzer Ausblick zur Einbindung der RSA-Verschlüsselung im schulischen Sekundarstufen-Bereich.

Inhaltsverzeichnis

1 Überblick: Das Prinzip der RSA-Verschlüsselung

1.1 Klassifizierung des Verfahrens in bekannte Kategorien

In einer zunehmend digitalisierten Welt nimmt die sichere Übermittlung sensibler Daten eine immer größere Rolle ein. Ein seit 1977 bewährtes, da überaus effizientes Verfahren stellt hierbei die RSA[1]-Verschlüsselung dar. Die grundlegende Idee basiert auf der mathematischen Tatsache, dass es zwar recht einfach ist, zwei große Primzahlen zu multiplizieren, die Rückrichtung, das heißt die Faktorisierung des Produkts, aber einen riesigen Zeitaufwand erfordert.
Zur besseren Einordnung des hier dargestellten Verfahrens in bereits bekannte Konzepte der Kryptographie sollen zu Beginn dieses Handouts die grundlegenden Prinzipien von RSA überblicksartig vorgestellt werden. Die nachfolgende Auflistung basiert zum größten Teil auf der Zusammenstellung von Schüller et al. [5]:

- Das RSA-Verfahren ist weder monoalphabetisch noch symmetrisch. Wie bei allen asymmetrischen Kryptosystemen werden Ver- und Entschlüsselung mit unterschiedlichen Schlüsseln durchgeführt.
- Der Schlüssel besteht aus zwei Teilen, einem öffentlichen und einem privaten.
- Der öffentliche Teil des Schlüssels wird zum Verschlüsseln der zu versendenden Nachricht verwendet. Er reicht aber nicht aus, um eine verschlüsselte Nachricht wieder zu entschlüsseln.
- Eine Entschlüsselung ist mithilfe des privaten Teils des Schlüssels trotzdem einfach möglich.
- Das RSA-Verfahren ist sehr sicher, denn man kann die Vorschrift, wie man die Nachricht wieder entschlüsseln kann, nicht oder nur mit einem sehr großen Zeitaufwand (in der Größenordnung von Jahren, selbst wenn man moderne Hochleistungsrechner zu Hilfe nimmt) ermitteln.
- Außerdem kann dieses Verfahren auch als digitale Signatur verwendet werden [2, S. 116]. Dabei gilt $\text{sig} := m^d \mod n$.

[1] Benannt wurde das Verfahren nach seinen Haupterfindern Ron Rivest, Adi Shamir und Leonard Adleman.

1.2 Allgemeine Funktionsweise

Bob	Alice
Erstellen der Schlüssel	
Man wählt zwei große Primzahlen p und q berechnet $n = p \cdot q$ berechnet $\phi(n) = (p-1)(q-1)$ (eulersche Phi-Funktion) wählt e teilerfremd zu $\phi(n)$ und bestimmt d mit $ed \equiv 1 \pmod{\phi(n)}$ Geheime Parameter bei der Schlüsselerzeugung: $p, q, \phi(n)$ Privater Schlüssel von Bob: d Öffentlicher Schlüssel von Bob: (e, n)	
Verschlüsselung	
	Wählt einen Klartext m und verwendet Bobs öffentlichen Schlüssel (n, e), um $c \equiv m^e \pmod{n}$ zu berechnen. Verschickt den Kryptotext c an Bob.
Entschlüsselung	
Verwendet d und berechnet damit $m' \equiv c^d \pmod{n}$. Dann entspricht m' dem Klartext m	

Abbildung 1.1: Aus: Beutelspacher et al. [2, S. 115] und Hoffstein et al. [3, S.123]

1.3 Konkretes Beispiel

Erstellen der Schlüssel
Bob wählt zwei Primzahlen $p = 71$ und $q = 83$, berechnet $n = 5893$ und $\phi(n) = 5740$, wählt $e = 17$ teilerfremd zu 5740 und bestimmt $d = 1013$ mit $ed \equiv 1 \pmod{5740}$. Geheime Parameter bei der Schlüsselerzeugung: 71; 83; 5740 Privater Schlüssel von Bob: 1013 Öffentlicher Schlüssel von Bob: 17; 5893
Verschlüsselung
Alice wählt einen Klartext m ≡ super mit $a \coloneqq 01; b \coloneqq 02; \ldots z \coloneqq 26$ s u p e r ≡ 19 21 16 05 18 ⇒ 1921 1605 1800 Sie verwendet Bobs öffentlichen Schlüssel (5893,17) und berechnet blockweise $1921^{17} \bmod 5893 \equiv 1172 \quad 1605^{17} \bmod 5893 \equiv 5791 \quad 1800^{17} \bmod 5893 \equiv 3536.$ Sie verschickt den Kryptotext $c \equiv 1172\ 5791\ 3536$ an Bob.
Entschlüsselung
Bob verwendet d = 1013 und berechnet $1172^{1013} \bmod 5893 \equiv 1921 \quad 5791^{1013} \bmod 5893 \equiv 1605 \quad 3536^{1013} \bmod 5893 \equiv 1800$ Es ergibt sich $m \equiv 19\ 21\ 16\ 05\ 18 \equiv$ s u p e r

Abbildung 1.2: Aus: Online-Resource: https://studyflix.de/informatik/rsa-verschlusselung-1608 (zuletzt aufgerufen am 07.06.2021).

2 Einblick: Die mathematischen Grundlagen

2.1 Einleitende Überlegungen

Wie man dem Überblick entnehmen kann, steht und fällt das RSA-Verfahren mit der Auswahl von passenden Primzahlen, der Bestimmung der Exponenten sowie einer verlässlichen Grundlage, dass die Ver- und Entschlüsselung auch in jedem Fall gelingt.

2.2 Primzahlen

Definition 2.2.1 Ein $p \in \mathbb{N}$ mit $p > 1$ nennt man *Primzahl*, falls 1 und p die einzigen (positiven) Divisoren von p sind. [4]

Für gewöhnlich verwendet man zur Verschlüsselung mit dem RSA-Verfahren Primzahlen der Größenordnung 2^{1024} [2, S. 118].
Zur Klärung der Frage, wie viele geeignete Zahlen sich in diesem Zahlenbereich befinden, hilft der Primzahlsatz.

Definition 2.2.2 Es sei $\mathbb{P}$ die Menge der Primzahlen. Für jede beliebige Zahl $x \in \mathbb{N}$ liefert $\pi(x) := |\{p \in \mathbb{P} \mid p \leq x\}|$ die Anzahl der Primzahlen bis x.

Beispiel 2.2.3 Es gilt $\pi(10) = 4$, da die Primzahlen zwischen 2 und 10 die vier Zahlen $2, 3, 5, 7$ sind.

Satz 2.2.4 (Primzahlsatz) Für alle $x \in \mathbb{N}$ gilt

$$\lim_{x \to \infty} \frac{\pi(x)}{\frac{x}{\ln(x)}} = 1$$

Beweis: [1, vgl. Chapter 13]. □
Damit wurde ein Ansatz zur Approximation für die Anzahl der Primzahlen bis zu einer bestimmten Größenordnung gefunden. Dieser lautet: $\pi(x) \approx \frac{x}{\ln x}$.

Beispiel 2.2.5 Es bezeichne $\pi(x)$ mit $x \in \mathbb{N}$ die Anzahl der Primzahlen, welche kleiner sind als x. Damit erhält man für Primzahlen mit genau 1024 Bit:
$\pi(2^{1024}) - \pi(2^{1023}) \approx \frac{2^{1024}}{\ln 2^{1024}} - \frac{2^{1023}}{\ln 2^{1023}} = \frac{2^{1024}}{1024 \cdot \ln 2} - \frac{2^{1023}}{1023 \cdot \ln 2} \approx \frac{2^{1023}}{1024 \cdot \ln 2} \approx 2^{1013}$.

Nachdem nun sichergestellt ist, dass genügend Primzahlen in der gewünschten Größenordnung existieren, stellt sich die Frage, wie man diese Primzahlen möglichst effizient findet. Hierzu gibt es diverse Primzahltests. Bei all diesen Ansätzen ist allerdings zu beachten, dass hierdurch keine Primzahlen generiert (Konstruktionsprinzip), sondern lediglich gegebene Zahlen auf die Eigenschaft „prim“ getestet werden (sowohl deterministisch als auch probabilistisch). Zu den wohl bekanntesten dieser Tests zählen *das Sieb des Eratosthenes* und der *Miller-Rabin-Test*.

Die Sicherheit des RSA-Verfahrens beruht auf dem Prinzip, dass das Produkt n zweier Primzahlen p und q im öffentlichen Schlüsselteil zugänglich gemacht wird, jedoch die Faktoren selbst geheimgehalten werden. Deshalb müssen Primzahlen gewählt werden, deren Produkt besonders schwer zu faktorisieren ist. Bei der Auswahl gilt es zu beachten, dass es Faktorisierungsalgorithmen wie *dem Pollard's-Rho-Algorithmus, dem Zahlkörpersieb* oder *dem Quadratischen Sieb* besonders schwer gemacht wird. Hieraus leiten Beutelspacher et al. [2, S. 118] folgende Anforderungen an die Primzahlen ab:

1. Je größer die Primzahlen, desto schwieriger die Faktorisierung des Produkts.
2. Die Primzahlen p und q sollten sich nicht zu stark unterscheiden, denn sonst arbeitet das Zahlkörpersieb zu effizient.
3. Die Primzahlen sollten aber auch nicht zu nahe beieinander liegen, denn andernfalls kann man mittels der Fermat-Faktorisierung effizient faktorisieren.
4. $(p+1)$ und $(p-1)$ sollten möglichst wenig „kleine“ Primteiler haben. Andernfalls ist Pollard's-$(p+1)$- bzw. $(p-1)$-Algorithmus sehr effizient.

Die in diesem Unterkapitel betrachteten Resultate stellen also sicher, dass im gewünschten Zahlenbereich hinreichend viele Primzahlen existieren und bieten einen Ansatz dafür, wie diese gefunden beziehungsweise auf ihre Eignung untersucht werden können.

2.3 Exponenten

Bei der Wahl der Exponenten sollte der Rechenaufwand zur Bestimmung der Potenzen berücksichtigt werden. Hierzu können Strategien wie beispielsweise der *Square-and-Multiply-Algorithmus* verwendet werden (vgl. [2, S. 119]). Insgesamt ist festzuhalten, dass die Verschlüsselung mit einem kleinen Verschlüsselungsexponenten effizienter ist, während dasselbe für die Entschlüsselung mit einem kleinen Entschlüsselungsexponenten gilt. Aufgrund ihrer gegenseitigen Abhängigkeit können nicht beide Exponenten gleichzeitig klein gewählt werden. Folglich muss bereits beim Erstellen der Schlüssel entschieden werden, welche Richtung einfacher gestaltet werden soll (vgl. [3, S. 125]). Zur Bestimmung des Entschlüsselungsexponenten d, also zum Lösen der Gleichung $e \cdot d \equiv 1 \pmod{n}$, verwendet man den *erweiterten euklidischen Algorithmus* (EEA) mit *Vielfachsummendarstellung* (vgl. [2, S. 120]).

2.4 Satz von Euler

Die grundlegende Gleichung, auf welcher das RSA-Verfahren basiert, ist die Verschlüsselung der Nachricht durch Anwendung des Verschlüsselungsexponenten und Modulo-Betrachtung des Ergebnisses, d.h. $x^e \equiv c \pmod{n}$. Hierbei bezeichnet x den Klartext, e den Verschlüsselungsexponenten, c den Kryptotext und n das Produkt der beiden Primzahlen. Als Vorbereitung zur Lösungsbetrachtung dieses Problems dienen die folgenden Prämissen:

Proposition 2.4.1 (i) Sei $n \geq 1$ ganzzahlig und sei $e \in \mathbb{Z}$. Dann existiert genau dann ein $d \in \mathbb{Z}$ mit $e \cdot d \equiv 1 \pmod{n}$, wenn $\mathrm{ggT}(e, n) = 1$.
(ii) Falls $e \cdot d_1 \equiv e \cdot d_2 \equiv 1 \pmod{n}$, dann gilt $d_1 \equiv d_2 (\mathrm{mod}\ n)$ und wir nennen d das multiplikative Inverse von e modulo n.

Beweis:
(i) „$\Leftarrow$ “Zunächst nehmen wir an, dass $\mathrm{ggT}(e, n) = 1$. Mit dem EEA wissen wir, dass $u, v \in \mathbb{Z}$ zu finden sind, für die gilt: $eu + nv = 1$. Daraus folgt: $eu - 1 = -nv$ und somit gilt: $n \mid (eu - 1)$, also $eu \equiv 1 \pmod{n}$. Wir können also $d = u$ wählen.

„⇒ “Angenommen e besitze eine Inverse modulo n, also $e \cdot d \equiv 1 \pmod{n}$. Damit gilt: $ed - 1 \equiv cn$ für ein $c \in \mathbb{Z}$. Es folgt, dass $\text{ggT}(e, n) \mid (ed - cn) = 1$ und somit $\text{ggT}(e, n) = 1$. Damit wurde also gezeigt, dass e genau dann eine (multiplikative) Inverse modulo n besitzt, wenn $\text{ggT}(e, n) = 1$ ist.
(ii) Nun bleibt noch zu zeigen, dass diese Inverse eindeutig ist modulo n. Dazu nehmen wir an, dass gilt: $e \cdot d_1 \equiv e \cdot d_2 \equiv 1 \pmod{n}$. Dann erhalten wir:
$d_1 \equiv d_1 \cdot 1 \equiv d_1 \cdot (e \cdot d_2) \equiv (d_1 \cdot e) \cdot d_2 \equiv 1 \cdot d_2 \equiv d_2 \pmod{n}$. Somit wurde auch die Eindeutigkeit gezeigt.

□

Proposition 2.4.2 Sei $p \in \mathbb{P}$ eine Primzahl und $e \geq 1$ ganzzahlig. Weiterhin gelte $\text{ggT}(e, p-1) = 1$. Mit Proposition 2.4.1 gilt, dass e eine Inverse modulo $p-1$ hat. Das heißt, es existiert genau ein d mit $de \equiv 1 \pmod{p-1}$.
Unter diesen Voraussetzungen hat die Äquivalenz $x^e \equiv c \pmod{p}$ die eindeutige Lösung $x \equiv c^d \pmod{p}$.

Beweis: Wir führen eine Fallunterscheidung durch:
Fall 1: Wenn $c \equiv 0 \pmod{p}$, dann ist $x \equiv 0 \pmod{p}$ die eindeutige Lösung und wir sind fertig.
Fall 2: Wenn $c \not\equiv 0 \pmod{p}$ ist, dann ist der Beweis eine Anwendung von Fermats kleinem Satz. Die Kongruenz $de \equiv 1 \pmod{p-1}$ bedeutet, dass es ein $k \in \mathbb{Z}$ gibt, so dass gilt: $de = 1 + k(p-1)$.
Wir prüfen nun, dass c^d eine Lösung für $x^e \equiv c \pmod{p}$ ist.

$$
\begin{aligned}
(c^d)^e &\equiv c^{de} \pmod{p} && \text{Potenzgesetze: Potenzieren von Potenzen} \\
&\equiv c^{1+k(p-1)} \pmod{p} && \text{mit oben: } de = 1 + k(p-1) \\
&\equiv c \cdot (c^{p-1})^k \pmod{p} && \text{Potenzgesetze} \\
&\equiv c \cdot 1^k \pmod{p} && \text{mit Fermats kleinem Satz} \\
&\equiv c \pmod{p}
\end{aligned}
$$

Hiermit wurde gezeigt, dass $x = c^d$ eine Lösung zu $x^e \equiv c \pmod{p}$ ist.
Nun wird noch die Eindeutigkeit gezeigt. Hierzu nehmen wir an, dass x_1 und x_2

beide Lösungen für die Kongruenz $x^e \equiv c \pmod{p}$ sind. Da wir gezeigt haben, dass $z^{de} \equiv z \pmod{p}$ für einen beliebigen Wert $z \neq 0$ gilt, finden wir, dass weiterhin gilt: $x_1 \equiv x_1^{de} \equiv (x_1^e)^d \equiv c^d \equiv (x_2^e)^d \equiv x_2^{de} \equiv x_2 \pmod{p}$.
Somit wurde gezeigt, dass x_1 und x_2 dasselbe modulo p sind, sodass die obige Kongruenz allenfalls eine Lösung hat.

□

Satz 2.4.3 (Satz von Euler)
Seien p und q verschiedene Primzahlen und $g = \text{ggT}((p-1),(q-1))$. Dann gilt: $a^{(p-1)(q-1)/g} \equiv 1 \pmod{pq}$ für alle a mit $\text{ggT}(a, pq) = 1$.

Beweis: Wir nehmen an, dass $p \nmid a$ und $g \mid (q-1)$. Dann gilt:

$$\begin{aligned} a^{(p-1)(q-1)/g} &= \left(a^{(p-1)}\right)^{(q-1)/g} && \text{denn } (q-1)/g \in \mathbb{Z} \\ &\equiv 1^{(q-1)/g} \pmod{p} && a^{(p-1)} \equiv 1 \pmod{p} \text{ Fermats kleiner Satz} \\ &\equiv 1 \pmod{p} && 1^n = 1 \; \forall n \in \mathbb{Z} \end{aligned}$$

Eine analoge Betrachtung gilt für den Fall, dass $q \nmid a$ und $g \mid (p-1)$. Es ergibt sich $a^{(p-1)(q-1)/g} \equiv 1 \pmod{q}$.
Damit ist gezeigt, dass $a^{(p-1)(q-1)/g} - 1$ durch p und durch q teilbar ist und damit teilbar durch pq.

□

Proposition 2.4.4 Seien p und q verschiedene Primzahlen und $e \geq 1$ mit $\text{ggT}(e,(p-1)(q-1)) = 1$. Mit Proposition 2.4.1 wissen wir, dass e genau ein multiplikatives Inverses modulo $(p-1)(q-1)$ besitzt, sodass $de \equiv 1 \pmod{(p-1)(q-1)}$ gilt. So hat die Kongruenz $x^e \equiv c \pmod{pq}$ die eindeutige Lösung $x \equiv c^d \pmod{pq}$.

Beweis: Wir nehmen an, dass $\text{ggT}(c, pq) = 1$.[2] Der Beweis verläuft fast identisch zum Beweis von Proposition 2.4.2, nur dass hier der Satz von Euler (Satz 2.4.3) angewendet wird und die Kongruenz $de \equiv 1 \pmod{(p-1)(q-1)}$ bedeutet, dass es ein $k \in \mathbb{Z}$ gibt, sodass $de = 1 + k(p-1)(q-1)$.

[2] Für die anderen Fälle verweise ich hiermit auf [3, S. 180].

Die Eindeutigkeit der Lösung zeigt sich daran, dass $x = u$ eine Lösung für $x^e \equiv c \pmod{pq}$ ist. Es gilt:

$$
\begin{aligned}
u &\equiv u^{de-k(p-1)(q-1)} \pmod{pq} && \text{denn: } de = 1 + k(p-1)(q-1) \\
&\equiv (u^e)^d \cdot (u^{(p-1)(q-1)})^{-k} \pmod{pq} && \\
&\equiv (u^e)^d \cdot 1^{-k} \pmod{pq} && \text{Satz von Euler} \\
&\equiv c^d \pmod{pq} && u \text{ ist eine Lösung für obige Kongruenz}
\end{aligned}
$$

Da jede Lösung zur obigen Kongruenz äquivalent zu $c^d \pmod{pq}$ ist, ist die Lösung eindeutig.

□

Beispiel 2.4.5
a) Seien $e = 5$ und $n = 21$ Bestandteile des öffentlichen Schlüssels, sowie der Kryptotext $c = 16$. Zum Entschlüsseln muss die Gleichung $x^5 \equiv 16 \pmod{21}$ gelöst werden. Mit Proposition 2.4.4 wissen wir, dass diese Gleichung die eindeutige Lösung $x \equiv 16^d \pmod{21}$ hat. Da die Primfaktorzerlegung von $21 = 3 \cdot 7$ offensichtlich ist, ergibt sich für $d = 5$ (denn: $d \cdot 5 \equiv 1 \pmod{12}$ wird gelöst mit $d = 5$) und damit für den Klartext: $x \equiv 16^5 \pmod{21} = 4$.

b) Der Schwierigkeitsgrad beim Lösen der zentralen Gleichung dieses Verfahrens erhöht sich mit der Größe der gewählten Primzahlen sowie der Länge der zu verschlüsselnden Nachricht. Betrachten wir nun noch einen Fall, in dem die Primfaktorzerlegung nicht offensichtlich ist. Seien $e = 17$, $n = 5893$ und $c = 1172$. Zum Entschlüsseln muss die Gleichung $x^{17} \equiv 1172 \pmod{5893}$ gelöst werden. Ohne Kenntnis des Entschlüsselungsexponenten d beziehungsweise der Primzahlen p und q (mit denen man über $d \cdot e \equiv 1 \pmod{(p-1)(q-1)}$ d bestimmen kann), ist kein Verfahren bekannt, mit dem man Gleichungen dieser Art effizient und mit vertretbarem Zeitaufwand lösen könnte. Sobald allerdings $d = 1013$ oder die Primzahlen $p = 71$ und $q = 83$ bekannt sind, ergibt sich der Klartext direkt aus $x = 1172^{1013} \pmod{5893} \equiv 1921$.

Zusammenfassend ist festzuhalten, dass der Satz von Euler die Grundlage für die korrekte Funktionsweise des RSA-Algorithmus liefert, da er die Existenz und die Eindeutigkeit der Lösung zur Verschlüsselungsgleichung $x^e \equiv c \pmod{n}$ garantiert. Die Betrachtung der mathematischen Grundlagen hat außerdem gezeigt, dass die Lösung der dem RSA-Verfahren zugrundeliegenden Gleichung sehr einfach berechnet werden kann, sofern n eine Primzahl ist oder man die Primfaktorzerlegung von n kennt. Hierzu liefert nämlich Proposition 2.4.4 direkt einen Lösungsalgorithmus. Es ist also im Sinne der Sicherheit des Verschlüsselungsverfahrens unbedingt darauf zu achten, dass n nur schwer zu faktorisieren ist beziehungsweise, dass die Primzahlen p und q aus dem privaten Schlüssel geheim gehalten werden (vgl. [3, S. 119f]).

3 Ausblick: Das RSA-Verfahren in der Schule

Gerade für Lehramtsstudenten ist das RSA-Verfahren auch deshalb interessant, da die dahintersteckende Mathematik recht gut didaktisch reduziert werden kann und die grundlegenden mathematischen Prinzipien bereits für Schülerinnen und Schüler der Mittelstufe nachvollziehbar aufgearbeitet werden können. Ein bereits didaktisch aufgearbeitetes Beispiel mit Hintergrundinformationen und Schüler-Arbeitsblättern findet sich in [5].

4 Bibliographie

Literatur

[1] Apostol, T.M., ‚Introduction to Analytic Texts in Mathematics', *Springer New York* (1976).

[2] Beutelspacher, A., Neumann, H. B., Schwarzpaul, T., ‚Kryptografie in Theorie und Praxis: Mathematische Grundlagen für Internetsicherheit, Mobilfunk und elektronisches Geld – 2.Auflage', *Vieweg+Teubner Wiesbaden* (2010).

[3] Hoffstein, J., Pipher, J., Silverman, J. H., ‚An Introduction to Mathematical Cryptography – 2nd Ed.', *Springer New York* (2014).

[4] Kuhlmann, S., ‚Gesamtscript zur Vorlesung Lineare Algebra I', *Konstanz* (2019).

[5] Schüller, A., Trottenberg, U., Wienands, R., Koziol, M., Schneider, R., ‚RSA – Primzahlen zur Verschlüsselung von Nachrichten', *Fraunhofer-Institut / Universität zu Köln* (2017).

[6] Online-Ressource (Video): 'RSA-Verschlüsselung' (https://studyflix.de/informatik/rsa-verschlusselung-1608 [zuletzt aufgerufen am: 07.06.2021]).