Versuch C: Verschlüsselung

Lara Quitte, Tino Jeromin

1 GnuPG

1.1 Einwegfunktionen/Hashfunktionen

1.1.1 Notizen zur Durchführung

- **Welche Hashfunktionen werden unterstützt?**
  SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
- **Berechnen Sie einen Hashwert einer Datei mittels einer Hashfunktion Ihrer Wahl.**
  Vorgehen: 'gpg –print-mds dateiname' (gibt alle Hashwerte an, mit 'gpg –print-md SHA1 dateiname' kann man die Hashfunktion auswählen)
- **Erläutern Sie die von Ihnen ausgewählte Hashfunktion näher. Welche Unterschiede treten auf in Bezug auf andere Hashfunktionen?**
  Die als Secure Hash Standard (SHS) bezeichnete Norm spezifiziert den sicheren Hash-Algorithmus(SHA) mit einem Hash-Wert von 160 Bit Länge für beliebige digitale Daten von maximal 264-1Bit Länge. SH1 wird heutzutage nicht mehr als sicher bezeichnet, da nicht kollisionsresistent ist. Der Hashwert hat bei unterschiedlichen Funktionen eine unterschiedliche Länge.
- **Laden Sie eine Datei aus dem Internet, für die ein Hashwert zur Verfügung gestellt wird, und überprüfen Sie den mitgelieferten Hashwert. Was kann man aus der Gleichheit schließen? Und was kann nicht aus der Gleichheit schließen?**
  Wenn man von Kollisionsfreiheit ausgeht, dann ist...
  - Hashwert unabhängig vom verwendeten Gerät und OS
  - bei Verwendung der selben Hashfunktion der Hashwert bei der Datei immer gleich
- **Welche der oben genannten Hashfunktionen sollte auf keinen Fall für den Zweck benutzt werden, sicherzustellen, dass die korrekte Datei ausgeliefert wurde?** MD5 sollte nicht genutzt werden, da dieses Verfahren veraltet ist und eine geringe Geschwindigkeit aufweist. Insbesondere hinsichtlich Kollisionsangriffen ist die Funktion nicht mehr ausreichend.

1.1.2 Hashfunktionen

Hashfunktionen bilden eindeutigen digitalen Fingerabdruck von Daten. Eine Hash-Funktion H ist eine Funktion die eine (variabel lange) Nachricht M auf einen Hash-Wert H(M) konstanter Länge abbildet.
Hash-Funktionen haben die folgenden Eigenschaften

- Einweg-Funktion: H ist nicht invertierbar, d.h. bei gegebenem Hash-Wert H(M) kann M nicht ermittelt werden (d.h nicht mit realistischem Rechenaufwand leistbar)

- Kollisionsresistent: Es ist ebenfalls (praktisch) nicht möglich, ein paar verschiedener Nachrichten M und M' zu finden, für die gilt : H(M) = H(M')

MD5:
Thirdly, similar to messages, you can also generate different files that hash to the same value so using MD5 as a file checksum is 'broken'.
SHA512
Secure if used correctly, few attacks
RIPEMD160
Old, but unbroken.
Cryptographers generallly aim for 128 to 256 bits of security. RIPEMD only offers 160 / 2 = 80 bits due to the birthday bound, which approximately halves the amount of security offered.

1.1.3 Kollision

Eine Funktion (in diesem Zusammenhang fast immer eine Einwegfunktion) wird als kollisionsresistent bezeichnet, wenn es schwer ist, verschiedene Eingaben zu finden, die auf denselben Wert abgebildet werden. Insbesondere bei kryptographischen Hashfunktionen handelt es sich hierbei um eine übliche Anforderung, deren Bruch in der Regel als Bruch der kompletten Hashfunktion betrachtet wird.

1.1.4 Fragen

Welche wichtigen Eigenschaften haben Hashfunktionen?

- wenn Eingaben verschieden sind, ist auch der Hashwert unterschiedlich
- gleiche Eingaben führen immer zum selben Hashwert
- Einweg-Funktion
- Kollisionsresistent
- Schnelligkeit: Berechnung muss schnell erfolgen

Wofür werden Hashfunktionen genutzt?

- eindeutigen Fingerabdruck von Daten bilden
- Prüfsummen erstellen
- Digitale Signaturen
- Speichern von Passwörtern

1.2 symmetrische Verschlüsselung

1.2.1 Notizen zur Durchführung

- **Welche Algorithmen der symmetrischen Verschlüsselung beherrscht GnuPG?** IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256

  - einen Text symmetrisch verschlüsseln und in eine Datei schreiben: *gpg -c -o output message*
  - wieder entschlüsseln und in Standardeingabe schreiben: *gpg -d output.txt*
  - entschlüsselte Nachricht in eine Datei schreiben: *gpg -o decrypt.txt -d output.txt*
  - Vergleichen Sie die Größe der verschlüsselten und unverschlüsselten Datei. Was fällt auf und wie lässt es sich erklären? *Die verschlüsselte Datei ist kleiner. Dies ist daruch zu erklären, dass die Daten zuerst automatisch komprimiert und dann verschlüsselt werden.*

- **Verschlüsseln Sie eine Datei mit dem BLOWFISH-Algorithmus, wählen Sie ZIP als Kompressionalgorithmus und schreiben Sie die verschlüsselten Daten in eine Ausgabedatei.** BLOWFISH ist ein symmetrischer Blockverschlüsselungsalgorithmus mit einer Blocklänge von 64 Bit. Er garantiert eine fehlerfreie Umkehrung von Ver- und Entschlüsselung. Die Schlüssellänge kann zwischen 32 Bit und 448 Bit betragen. Aus diesen Schlüsselbits werden vor Beginn der Ver- oder Entschlüsselung die so genannten Rundenschlüssel P1 bis P18 und die Einträge in den S-Boxen erzeugt, insgesamt 4168 Byte.
  Generell zählen sowohl .zip als auch .rar zu den verlustfreien Kompressionsverfahren. Das heißt, dass die Datei nach dem Packen und anschließendem Entpacken zu 100 Prozent identisch mit dem Original wieder hergestellt wird.

  Anweisung: *gpg -e –cipher-algo BLOWFISH –compress-algo ZIP -o compressed decrypt*

- **Verschlüsseln Sie eine Datei mit dem IDEA-Algorithmus, wählen Sie BZIP2 als Kompressionalgorithnmus und schreiben Sie die verschlüsselten Daten in eine Ausgabedatei.** IDEA benutzt eine Serie von acht identischen Transformationen, welche je einer Runde entsprechen, und einer Ausgabetransformation, welche einer halben Runde entspricht. Der Entschlüsselungsprozess entspricht dem Verschlüsselungsprozess in umgekehrter Form. Bei der Verschlüsselung wird der Klartext in 64 Bit große Blöcke unterteilt und der Schlüssel in Teilstücke zu je 16 Bit zerlegt.
  Bzip2 ist ein eigener Kompressionsalgorithmus. Er benutzt eine Kombination von verschiedenen Methoden, um die Datei zu komprimieren. Zuerst werden die Daten mit der Burrows-Wheeler-Transformation und der Move-To-Front-Transformation transformiert, wobei die Dateigröße sich noch nicht verändert. Danach wird die Datei mit der Run-Length- Kodierung und der Huffman-Kodierung komprimiert.

  Anweisung: *gpg -e –cipher-algo IDEA –compress-algo BZIP2 -o compressed decrypt*
  Diese Datei ist etwas größer als die mit ZIP.

1.2.2 symmetrischer Schlüssel

- Beide Parteien haben den gleichen Schlüssel
- Wird zum Ver- und Entschlüsseln benutzt
- Je länger der Schlüssel, desto sicherer
- Schneller als asymmetrisch
- Für jeden Kommunikationspartner ein Schlüssel - sehr viele Schlüssel
- symmetrische Verschlüsselung gewährleistet:

  - Geheimhaltung – weil eine verschlüsselte Nachricht nicht ohne Kenntnis des geheimen Schlüssels dechiffriert werden kann
  - Integrität – weil eine verschlüsselte Nachricht nicht ohne Kenntnis des geheimen Schlüssels so modifiziert werden kann, dass die Dechiffrierung normalen Klartext ergibt
  - Authentizität – weil eine verschlüsselte Nachricht, deren Dechiffrierung normalen Klartext liefert, nur von jemandem, der den geheimen Schlüssel kennt, stammen kann

- mögliches Problem: Integrität und Authentizität sind eventuell bedroht durch Wiedereinspielen (replay) zuvor abgehörter (verschlüsselter) Nachrichten oder Nachrichtenteile
- mögliche Lösung: Nachrichten durchnumerieren Rückkoppelung (gemäß Cipher Block Chaining, Output Feedback, Cipher Feedback)Nachricht mit Zeitangabe versehen
- verbleibende Schwäche: Geheimer Schlüssel ist Gruppenschlüssel von 2 (oder mehr) Partnern
- Verbindlichkeit ist daher nicht gewährleistet: A: "Ich habe von B diese Nachricht erhalten" ist weder beweisbar noch widerlegbar.

1.2.3 symmetrische Ver- und Entschlüsselung von Dateien/Texten

- der selbe Schlüssel wird zur Ver- und Entschlüsselung genutzt
- will man einen symmetrisch verschlüsselten Text weitergeben, so muss man auch den Schlüssel mitgeben (öffentlicher Schlüssel)
- das eigentliche Problem der symmetrischen Verschlüsselung ist die Weitergabe des Schlüssels, da ein sicherer Kanal für die Übermittlung nötig ist - E-Mail fällt also z.B weg
- da die Weitergabe also nicht immer trivial, daher asymmetrische Verschlüsselung oder Hybrid sicherer
- symmetrische Verschlüsselung sinnvoll für lokal genutzte Dateien, da Schlüssel dann nicht weitergegeben werden muss
- man benötigt: Datei, Schlüssel, Algorithmus

1.3 Asymmetrische/hybride Verschlüsselung

1.3.1 Notizen zur Durchführung

- Welche Algorithmen der asymmetrischen Verschlüsselung beherrscht GnuPG? RSA, ELG, DSA, ECDH, ECDSA, EDDSA
- Erzeugen Sie dann pro Gruppenmitglied mindestens ein Schlüsselpaar von sinnvoller Art und Größe. Erklären Sie, warum die von Ihnen gewählte Schlüsselgröße sinnvoll ist (die Defaulteinstellung ist KEIN Argument). Erzeugen Sie außerdem ein Widerrufszertifikat.
  - gpg –gen-key Erstellt einen Schlüssel
  - gpg –output revoke.asc –gen-revoke 2183480FE2402B1A Erstellt ein Widerrufszertifikat zum jeweiligen Schlüssel

  - Die Schlüssellänge ist 3072 Bits, dies entspricht etwa der AES und ist damit als sicher einzustufen.

- Exportieren Sie dann jeweils Ihren öffentlichen Schlüssel im ASCII-Format und importieren den Ihres Gruppenmitglieds (an dieser Stelle sollten Sie sich klar machen, dass dieser Schlüsseltausch eine kritische Angelegenheit ist: Es sollte stets sichergestellt werden, dass der erhaltene Schlüssel tatsächlich zur richtigen Person gehört).

  - Exportieren: gpg –export -a [UID]
  - Importieren: gpg –import [Datei]

- Verschlüsseln Sie jeweils eine Datei für Ihr Gruppenmitglied (welchen Schlüssel nehmen Sie?). Machen Sie sich klar, was in welcher Reihenfolge auf welche Art verschlüsselt wird (dies soll sich in der eingangs erwähnten Skizze wiederfinden). Entschlüsseln Sie diese wieder.

  - gpg –encrypt -o encryptMessage –sign -r davidjer@hotmail.de -u 2183480FE2402B1A message
    Datei wird mit private key verschlüsselt
  - gpg encryptMessage
    verschlüsselte Datei wird Empfänger entschlüsselt

- Schicken Sie sich gegenseitig verschlüsselte E-Mails zu und lesen die entschlüsselte Nachricht des anderen. Wählen Sie dazu einen geeigneten E-Mail-Client und richten ihn entsprechend ein. Machen Sie sich klar, was in welcher Reihenfolge auf welche Art verschlüsselt wird.

  - gpg –export-secret-keys –armor stu225209@mail.uni-kiel.de>mein-key.asc
    Exportieren der secret keys um sie in thunderbird zu importieren

1.3.2 Stichwörter

- asymmetrische Verschlüsselung

  - Ein privater und ein öffentlicher Schlüssel für jeden Kommunikationspartner
  - Eine mit dem public key verschlüsselte Nachricht kann nur mit dem private key entschlüsselt werden und andersherum
  - asymmetrische Verschlüsselung ermöglicht durch die Verschlüsselung einer Nachricht mit dem öffentlichen Schlüssel die Vertraulichkeit bzw. Geheimhaltung der Nachricht, da der verschlüsselte Text nur mit dem privaten Schlüssel gelesen werden kann
  - Verschlüsselung mit dem öffentlichen Schlüssel garantiert jedoch nicht die Authentizität des Ursprungs der Nachricht, da der öffentliche Schlüssel von allen benutzt werden kann und allein an der Verschlüsselung nicht gesehen werden kann, von wem die Nachricht stammt

- Schlüsselpaar

  - Absender und Empfänger verwenden verschiedene Teile eines Schlüsselpaares
  - jeder Nutzer erzeugt sein eigenes Schlüsselpaar bestehend aus einem öffentlichen und einem privaten Schlüssel
  - der öffentliche Schlüssel ermöglicht es jedem, Daten für den Besitzer des privaten Schlüssels zu verschlüsseln, dessen digitale Signaturen zu prüfen oder ihn zu authentifizieren

  - der private Schlüssel ermöglicht es seinem Besitzer, mit dem öffentlichen Schlüssel verschlüsselte Daten zu entschlüsseln, digitale Signaturen zu erzeugen oder sich zu authentisieren

- Schlüsselring
  - dient zur sicheren Speicherung und Verwaltung von Passwörtern
  - Passwörter werden verschlüsselt gespeichert und mit einem Masterpasswort geschützt
  - Auslesen der Passwörter ist auch beispielsweise nach dem Diebstahl eines Notebooks nicht möglich
  - das Masterpasswort muss eine hohe Qualität haben und für niemanden zugänglich sein
- Schlüsselimport
- Schlüsselexport
- öffentlicher Schlüssel: ermöglicht es jedem, Daten für den Besitzer des privaten Schlüssels zu verschlüsseln, dessen digitale Signaturen zu prüfen oder ihn zu authentifizieren
- privater Schlüssel: ermöglicht es seinem Besitzer, mit dem öffentlichen Schlüssel verschlüsselte Daten zu entschlüsseln, digitale Signaturen zu erzeugen oder sich zu authentisieren
- Schlüsselserver: bietet Zugang zu öffentlichen Schlüsseln, die in asymmetrischen Kryptosystemen dazu benutzt werden, einer Person verschlüsselte Mitteilungen – beispielsweise per E-Mail – zu senden oder ihre Signaturen zu verifizieren
- Bestandteile eines Schlüssels
- asymmetrische Ver- und Entschlüsselung von Dateien/Nachrichtens

1.3.3 Fragen

Erklären Sie anhand Ihres Versuches zur verschlüsselten E-Mail, wie die hybride Verschlüsselung genau funktioniert.
Pretty Good Privacy nutzt eine Variante des Public-Key-Systems, bei dem jeder Anwender einen öffentlichen Schlüssel besitzt, der beliebig weitergegeben werden kann. Dazu kommt ein privater Schlüssel, der nur dem Anwender bekannt ist. Um eine verschlüsselte Nachricht an jemand anderes zu versenden, benötigen Sie seinen öffentlichen Schlüssel. Wenn der Empfänger die Nachricht erhalten hat, kann er sie mit seinem privaten Schlüssel decodieren. Weil es sehr zeitaufwändig sein kann, eine komplette Nachricht zu verschlüsseln, setzt PGP auf einen schnelleren Algorithmus. Dieser verwendet einen zusätzlichen kürzeren Key, um die Nachricht zu verschlüsseln. Nur dieser kürzere Schlüssel wird dann mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Sowohl die chiffrierte Nachricht als auch der verschlüsselte kürzere Schlüssel werden dann an den Empfänger gesendet. Mit seinem privaten Schlüssel kann dieser zunächst den verschlüsselten Key entschlüsseln, und diesen dann dazu verwenden, die eigentliche Nachricht wieder lesbar zu machen.
Unter Hybrider Verschlüsselung, auch Hybridverschlüsselung genannt, versteht man eine Kombination aus asymmetrischer Verschlüsselung und symmetrischer Verschlüsselung. Dabei wählt der Sender einen zufälligen symmetrischen Schlüssel, der Session-Key bzw. Sitzungsschlüssel genannt wird. Mit diesem Session-Key werden die zu schützenden Daten symmetrisch verschlüsselt. Anschließend wird der Session-Key asymmetrisch mit dem öffentlichen Schlüssel des

Empfängers verschlüsselt. Dieses Vorgehen löst das Schlüsselverteilungsproblem und erhält dabei den Geschwindigkeitsvorteil der symmetrischen Verschlüsselung.
Nach welchen Kriterien haben Sie Ihre Verschlüsselungsalgorithmen ausgesucht und die Schlüssellänge bestimmt?

1.4 Signaturen

Eine digitale Signatur, auch digitales Signaturverfahren, ist ein asymmetrisches Kryptosystem, bei dem ein Sender mit Hilfe eines geheimen Signaturschlüssels (dem Private Key) zu einer digitalen Nachricht (d. h. zu beliebigen Daten) einen Wert berechnet, der ebenfalls digitale Signatur genannt wird. Dieser Wert ermöglicht es jedem, mit Hilfe des öffentlichen Verifikationsschlüssels (dem Public Key) die nichtabstreitbare Urheberschaft und Integrität der Nachricht zu prüfen. Die digitale Signatur sichert die Echtheit des Senders, da kein anderer den geheimen Schlüssel zum Signieren besitzt. Die digitale Signatur sichert jedoch nicht die Vertraulichkeit der Nachricht, da jeder den öffentlichen Schlüssel des Signierenden bekommen kann, um damit die Nachricht zu lesen. Daher kann man eine signierte Nachricht noch zusätzlich mit dem öffentlichen Schlüssel des Empfängers verschlüsseln, um sowohl Vertraulichkeit als auch Authentizität zu erreichen.

1.4.1 Notizen zur Durchführung

Wie in der Einleitung erwähnt, ist die Echtheit eines öffentlichen Schlüssels die Achillesferse des Systems. Deshalb gibt es die Möglichkeit, Schlüssel zu unterschreiben. Damit bestätigt der Unterzeichnende, daß der in der User ID angegeben User tatsächlich der Besitzer des Schlüssels ist.
Nachdem man mit gpg –edit-key UID den zu unterzeichnenden Schlüssel ausgewählt hat, kann man ihn mit dem Kommando sign unterschreiben.
Unterschreiben Sie nur Schlüssel von deren Echtheit sie sich überzeugt haben. Das kann geschehen, in dem man entweder den Schlüssel persönlich bekommen hat (zum Beispiel auf einer Keysigning Party), oder man über Telefon den Fingerprint vergleicht. Man sollte keinen Schlüssel nur deshalb unterschreiben, weil man den anderen Unterschriften vertraut.
Anhand der Unterschriften und des "ownertrusts" ermittelt GnuPG die Gültigkeit des Schlüssels. Der Ownertrust ist ein Wert mit dem der Benutzer festlegt, in welchem Maße er dem Schlüsselinhaber zutraut, andere Schlüssel verläßlich zu unterzeichnen. Die möglichen Abstufungen sind "gar nicht", "weiß nicht", "teilweise" und "vollständig". Wenn der Benutzer also einem anderem nicht traut, kann er GnuPG über diesen Mechanismus anweisen, dessen Unterschrift zu ignorieren. Der Ownertrust wird nicht im Schlüsselbund gespeichert, sondern in einer separaten Datei.

- Schicken Sie jeweils dem anderen eine signierte Nachricht, die ansonsten unverschlüsselt ist und prüfen Sie die Ihnen zugesandte signierte Nachricht. Was passiert intern in welcher Reihenfolge? Macht es einen Unterschied, ob Sie –sign oder –clearsign benutzen?
  - gpg -s -o signed message (**Sign**) Hier wird die eigentliche Nachricht so verändert, dass sie nicht mehr in einem Editor geöffnet werden kann. Die Nachricht ist noch lesbar. Das Dokument wird komprimiert und der Output ist ein Binary.
  - gpg –clearsign -o clearsigned message (**ClearSign**) Hier wird Signatur an den Text angehängt. Die Nachricht bleibt erhalten. Die Signatur wird als ASCII Zeichen geschrieben.
  - wird die Nachricht zusätzlich vorher mit *gpg -s -e -r stu225209@mail.uni-kiel.de -o signedEncrypted message* verschlüsselt und dann signiert, so erhält man eine verschlüsselte Datei, die außerdem signiert ist

- das Verschlüsseln einer clearsigned message ist nicht möglich, da nicht sinnvoll
- Importieren öffentlicher Schlüssel
  gpg - -import Schlüssel (Nimmt öffentlichen Schlüssel Schlüssel ins Schlüsselbund auf)
  - mit save in gpg bestätigen
- Signieren des öffentlichen Schlüssels
  gpg - -edit-key UID - danach sign und save
- Mit *gpg –list-sig davidjer@hotmail.de* kann man prüfen, wer dem angegebenen Schlüssel vertraut
- wird bei einer signierte Mail der Text verändert und ohne weitere Änderungen an der Signatur verschickt, so ist die Signatur am Ende nicht mehr vorhanden
- Verschlüsseln Sie eine Datei mit AES256 und signieren Sie diese (dieses sollte in einem einzigen Befehl stattfinden
  gpg -c -o aesSigned –cipher-algo AES256 -s message

1.4.2 Stichwörter

- Signieren von Dateien/Nachrichten: Sicherstellung, dass eine bestimmte Datei/Nachricht von einem bestimmten Absender stammt (Authentizität) und nicht manipuliert wurde (Integrität) - Signaturen werden mit privaten Schlüsseln erstellt und es kann jede Art von Datei signiert werden
- Bestandteile eines Schlüssels: mehrere Eigenschaften wie Name und E-Mail des Eigentümers, den Schlüsseltyp, das Ablaufdatum usw.
- Fingerprint eines Schlüssels: eindeutige Folge von Buchstaben und Zahlen, mit welcher der Schlüssel identifiziert werden kann, wenn Eigenschaften mehrerer Schlüssel gleich sind
- Signieren von Schlüsseln: man muss sich von der Echtheit des verwendeten Schlüssels zu überzeugen - dies geschieht mittels des Fingerprints oder einer Signatur

1.4.3 Fragen

Welches Ziel verfolgt man bei der Signierung eines Schlüssels? Wie wird dieses Ziel technisch realisiert? Die digitale bzw. elektronische Signatur ist eine schlüsselabhängige Prüfsumme, die von einer Nachricht oder einem Dokument in Kombination mit einem Schlüssel erzeugt wird. Wird die Signatur an eine Nachricht oder ein Dokument angehängt, dann gilt das als unterschrieben. Für digitale Nachrichten und Dokumente werden digitale Signaturen verwendet, um ihre Echtheit glaubhaft und prüfbar zu machen. Die Echtheit der Signatur kann elektronisch geprüft werden.
Um Nachrichten und Dokumente zu signieren muss dessen Ersteller die Nachricht mit seinem privaten Schlüssel "entschlüsseln". Der dabei entstandene "Klartext" ist die digitale Signatur. Sie wird an das Dokument angehängt. Die Signatur kann von jedem anderen mit dem öffentlichen Schlüssel des Erstellers "verschlüsselt" werden. Dabei entsteht die ursprüngliche Nachricht, die mit der unverschlüsselten Nachricht verglichen werden kann. Sind beide gleich, ist das Dokument unverändert und korrekt signiert.
RSA als Signaturverfahren: Ein Teilnehmer "unterschreibt" eine Nachricht m, indem er sie mit seinem privaten Schlüssel d kodiert. Heraus kommt die Signatur s. Er verschickt die Nachricht m zusammen mit der Signatur s. Die Echtheit der Nachricht m und die Identität der Person kann durch die Signatur s und den öffentlichen Schlüssel von jedem überprüft werden.

Wenn ein Benutzer ein Dokument elektronisch unterschreibt, wird unter Nutzung des privaten Schlüssels des Unterzeichners eine Signatur erzeugt. Der private Schlüssel wird vom Unterzeichner geheim gehalten. Der mathematische Algorithmus arbeitet wie eine Chiffre und erzeugt Daten zu dem betreffenden Dokument, Hash genannt, und verschlüsselt die Daten. Die resultierenden verschlüsselten Daten sind die digitale Signatur. Die Signatur wird zudem mit einem Zeitstempel versehen. Wenn das Dokument nach der Unterzeichnung verändert wird, ist es ungültig.

2 Festplattenverschlüsselung

2.1 Notizen zur Durchführung

- **Erstellen Sie einen verschlüsselten Container als Datei auf der Festplatte ihrer Virtuellen Maschine. Wählen Sie ein kleine von nur einigen MB**
  Schlüssel zum entsperren des Containers :
  `dd if=/dev/urandom of=master.keyfile bs=4096 count=1`
  Leeren Container mit 16MG erstellen:
  `dd if=/dev/zero of=CONTAINER bs=1 count=0 seek=16MB`
  Container verschlüsseln:
  `sudo cryptsetup -y -c aes-xts-plain64 -s 512 -h sha512 -i 5000 --use-random luksFormat CONTAINER master.keyfile`
  Entsperren des Containers:
  `sudo cryptsetup luksOpen CONTAINER PRIVATE --key-file master.keyfile`
  Formatieren des entsperrten Mediums:
  `sudo mkfs.ext4 /dev/mapper/PRIVATE`

- **Mounten Sie den Container in Ihrem Dateisystem.**
  Mounten des entsperrten Mediums auf einen lokalen Ordner
  `sudo mount /dev/mapper/PRIVATE /media/sf_OneDrive/Informatik_CAU/IT_Security/C_Verschlsselung/Teil_2/MountedContainer/`
  Dateiberechtigungen einstellen:
  `sudo chown -R user1:user1 /dev/mapper/PRIVATE /media/sf_OneDrive/Informatik_CAU/IT_Security/C_Verschlsselung/Teil_2/MountedContainer/`

- **Legen Sie in diesem Container drei Dateien an. Schreiben Sie 'AAAAA' in die erste Datei, schreiben Sie 'BBBBB' in zweite Datei, etc.**
  Siehe Screenshot.

- **Unmounten Sie den Container.**
  `sudo umount/media/sf_OneDrive/Informatik_CAU/IT_Security/C_Verschlsselung/Teil_2/MountedContainer/`

2.2 verschlüsselte Container

Container sind insbesondere geeignet, auf einer ansonsten nicht verschlüsselten Partition einen privaten verschlüsselten Bereich für sensible Daten anzulegen. Innerhalb eines Containers verwaltet TrueCrypt ein Dateisystem. Zum Lesen und Schreiben mountet TrueCrypt diese Datei. Unter Windows wird dazu ein neues virtuelles Laufwerk erstellt. Unter macOS und Linux wird der Container in ein beliebiges Verzeichnis eingehängt. Zugriffe auf das Laufwerk/das Verzeichnis unterscheiden sich nicht von Zugriffen auf andere, nicht durch TrueCrypt erzeugte Pendants.

Die Ver- und Entschlüsselung übernimmt der TrueCrypt-Treiber im Hintergrund (englisch on the fly). Container können, wenn sie nicht eingebunden sind, wie normale Dateien behandelt werden, beispielsweise auf eine DVD gebrannt werden.

2.2.1 Fragen

Betrachten Sie den Inhalt des Containers (also der Datei in ihrem Dateisystem und nicht der gemounteten virtuellen Festplatte) in einem Editor oder Kommandozeilenwerkzeig (cat, less, ...). Können Sie Informationen über die Dateien und ihre Inhalte finden? Nein.

3 SSH Schlüssel

3.1 Notizen zur Durchführung

- **Erstellen Sie ein SSH Schlüsselpaar.**
  `ssh-keygen`
- **Kopieren Sie den öffentlichen Schlüssel des Schlüsselpaares entsprechend der Anleitung (siehe Vorbereitung) auf Ihre virtuelle Maschine (oder einen anderen Server, wie zum Beispiel einen der SSH Server der Informatik. Eine Liste der Server finden Sie auch unter der Überschrift "SSH Server des IfI" auf den Seiten der Fachschaft), und ermöglichen Sie den Login mittels SSH Schlüssel. Zum Kopieren können Sie unter anderem scp nutzen.**
  `ssh-copy-id stu225209@boromir.informatik.uni-kiel.de`
- **Nutzen Sie Ihren privaten Schlüssel des Schlüsselpaares für einen Login. Es sollte also kein Passwort mehr für den Login benötigt werden, sondern der Schlüssel genutzt werden.**
  `ssh stu225209@boromir.informatik.uni-kiel.de`

3.2 Stichwörter

- SSH Schlüssel
  - Sicherer als Passwörter
  - Kein Passwort zum login nötig
  - Passwort wird nicht über das Netzwerk geschickt
  - Usernames and passwords are easily remembered, and if web login is possible, browsers can auto-fill these fields, making it even easier to log in.
  - Sichere Passwörter sind schwer zu merken und man muss sie regelmäßig ändern
- Agenten: Bei jedem SSH-Login mit Public-Key-Authentifizierung muss die Passphrase des jeweiligen privaten Schlüssels eingegeben werden. Wenn solche Verbindungen häufiger aufgebaut werden müssen, macht es Sinn einen SSH Key-Agent zu verwenden. Dieser hält alle privaten Schlüssel dekodiert im Speicher, sobald man einmal die entsprechende Passphrase eingegeben hat. Ein ständiges Eintippen der Passphrase für jede neue SSH-Verbindung entfällt somit.

3.3 Fragen

Was für Vorteile und Nachteile hat ein SSH Schlüssel gegenüber einem Passwort-basierten Login?
siehe oben

4 SSH Tunnel, Port Forwarding, SOCKS Proxy

4.1 Notizen zur Durchführung

- **Setzen Sie von Ihrem Rechner zu Hause (der keine CAU IP-Adresse hat) einen SSH Tunnel zu ihrer Virtuellen Maschine oder einem SSH Server der Informatik auf.**
  `ssh -N -D 9090 user1@134.245.253.153 -p 19116`

4.2 Stichwörter

- SSH Tunnel
  Tunnel meint nun, dass Daten von einem dritten Rechner, zum Beispiel eine Webseite, vom SSH-Server angefordert und dann über die SSH-Verbindung an Ihren lokalen Rechner weitergeleitet werden. Angenommen, Ihr Netzwerkadministrator hat die Webseite "example.com" gesperrt. Sie können aber auf Ihren SSH-Server "ssh.meinsshserver.de" zugreifen - und dieser wiederum auch auf "example.com". Dann sagen Sie Ihrem SSH-Server, dass er "example.com" an Ihren lokalen Rechner leiten soll - und Sie können sie dann im Browser ansehen. Das Beste: SSH-Verbindungen sind wie gesagt verschlüsselt, sprich es ist für niemanden ersichtlich, welche Daten, hier also die Webseite von "example.com", Sie sich anschauen.
  A Secure Shell (SSH) tunnel consists of an encrypted tunnel created through an SSH protocol connection. Users may set up SSH tunnels to transfer unencrypted traffic over a network through an encrypted channel.

- Port Forwarding
  - Local port forwarding: connections from the SSH client are forwarded via the SSH server, then to a destination server
  - Remote port forwarding: connections from the SSH server are forwarded via the SSH client, then to a destination server
  - Dynamic port forwarding: connections from various programs are forwarded via the SSH client, then via the SSH server, and finally to several destination servers

- SOCKS Proxies: Das SOCKS-Protokoll ist ein Internet-Protokoll. Dieses erlaubt Anwendungen, Daten protokollunabhänging über einen Proxyserver zu leiten. SOCKS-Proxies sind dadurch universell einsetzbar z.B. zum Abrufen von Webseiten, versenden von Mails, selbst die Daten von online Spielen lassen sich so mit speziellen Programmen (siehe weiter unten) über Proxyserver leiten.

4.3 Fragen

Was sind die Vorteile und Nachteile von SSH Tunnel gegenüber einem VPN Tunnel?

- VPN benötigt Client-software und einen VPN-Server

- SSH Tunnel ist einfacher einzurichten
- Anfänger können problemlos eine Verbindung zu einem VPN herstellen, das Einrichten eines VPN-Servers ist jedoch komplexer
- SSH-Tunnel ist der einfachste Weg, den Netzwerkverkehr zu verschlüsseln und zu tunneln
- bei beiden ist die Verschlüsselung gleich gut

5 Quellen

- https://www.dr-datenschutz.de/hashwerte-und-hashfunktionen-einfach-erklaert/
- http://www.inf.fu-berlin.de/lehre/SS04/SySi/folien/KryptografieTeil2.pdf
- https://de.wikipedia.org/wiki/Verschlüsselung
- https://help.gnome.org/users/seahorse/stable/misc-key-fingerprint.html.de
- https://graphentech.io/ge/vpn-vs-ssh-tunnel-was-ist-sicherer/
- https://www.elektronik-kompendium.de/sites/net/1910131.htm