Anhang: Seminarprogramm „Höhere Protokolle“

Als Skizze, mit welchen Themen man sich nach dieser Vorlesung beschäftigen kann, hier die Themenliste des Seminars „Kryptologie“ mit Schwerpunkt „Höhere Protokolle“ aus dem WS 94/95 (BetreuerInnen Gerrit Bleumer, Birgit Pfitzmann, Matthias Schunter, Joachim Biskup).

A. Analyse / Annahmen

Überblick: [LeLe_90, BachE_90].

A.1 Das quadratische Sieb — (noch) der beste Faktorisierungsalgorithmus in der Praxis.

Literatur: [Pome1_90]. Ergänzend: [Pome_85, Cohe1_93].

a) Einleitung: Was hat Faktorisierung mit kryptologischen Protokollen zu tun? Geschichtliche Entwicklung aus [Pome1_90].

b) QS Algorithmus verständlich darstellen (incl. Laufzeitangabe) und erklären, wieso er faktorisiert [Pome1_90 ohne §§5-6].

c) Erklärung des „multiple polynomial quadratic sieve“ [Pome1_90].

d) Evtl. Herleitung der Laufzeitabschätzung [Pome_84].

e) Zur Durchführung in der Praxis: Laufzeitvergleich mit Number Field Sieve [Pome1_90], Rechnerstruktur/Rechenzeit aus [DiLe_94, DDLM _94, LeMa1_90], RSA129 [News].

A.2 (p–1)-Verfahren: Faktorisierung und diskrete Logarithmen für Zahlen spezieller Form

a) Einleitung:
Für Zahlen n, die einen Primfaktor p haben, für den $p - 1$ aus lauter kleinen Primfaktoren besteht, existieren effiziente Faktorisierungsalgorithmen. Wie häufig sind diese Zahlen in der Theorie und in der Praxis [BrDL_93, S. 447]. Wie einfach ist es, sie zu vermeiden [Gord_85]?

b) Pollards (p–1)-Algorithmus zur Faktorisierung [Cohe1_93, S. 431ff.]. (Original: [Poll_74].)

c) Beschreibung des Pohlig-Hellman-Algorithmus für diskrete Logarithmen [PoHe_78].
Der Algorithmus wird dort nur für Gruppen $\mathbb{Z}_p^*$ beschrieben. Er sollte aber in allen Gruppen G funktionieren, wo $|G|$ nur kleine Primfaktoren hat. Es wäre gut, ihn gleich für diesen allgemeineren Fall darzustellen, weil es bedeutet, daß man bei Diskreter-Logarithmus-Kryptosystemen, ganz egal in welcher Gruppe, bzgl. dieses Algorithmus aufpassen muß.

B. Hashfunktionen

B.1 Hashfunktionen

Kommen hier in Protokollseminar einerseits vor, weil sie in beiden Vorlesungen (Kryptographie, Sicherheit in Rechnernetzen) wenig behandelt werden, aber wichtig sind, und andererseits unter dem Aspekt, daß man aus kleineren Primitiven größere macht.

Falls modular, geht man von einer "nicht invertierbaren" Permutation $\pi(k, s, b)$ aus, so daß die lineare Rekursion $f(s'||s, b_1) := f(s, \pi(k, s', b_1)$ eine vollständige Hashfunktion ergibt. Jetzt Doppelcharakter von f betrachten: $f(s, \bullet)$, mit erstem Parameter variabel: bündelnd und kollisionsfrei. $f(\bullet, b)$, mit zweitem Parameter variabel: Permutation, also auch invertierbar.

Worauf es ankommt, hängt von Anwendung ab. Betrachte Signatursysteme → Nichtinvertierbarkeit im zweiten Parameter, da erster z.B. bei chosen message attack bekannt, UND Kollisionsfreiheit im ersten Parameter erforderlich!

Drei Typen:

- **Einweg-Hashfunktionen**: es existiert kein Schlüsselparameter k. Sicherheit beruht auf Chaosannahme (z.B. DES). Ziemlich vollständiger Überblick in [PrGV_94]. Evtl. auch birthday-Angriffe aus [DaPr_89] ansehen.
- **Private Key Hashfunktionen**: Kenntnis von k ermöglicht nicht nur π, sondern auch π^{-1} effizient zu berechnen. Sicherheit beruht auf Chaosannahme (z.B. DES). [LaMa_93]
- **Public Key Hashfunktionen**: Kenntnis von k ermöglicht nur, π effizient zu berechnen. Zu jedem k existiert jedoch ein geheimer Schlüssel sk, mit dessen Kenntnis auch π^{-1} (Invertierung in b!) effizient berechnet werden kann. Sicherheit beruht auf komplexitätstheoretischer Annahme wie Faktorisierung oder diskretem Logarithmus. [Damg_88]

  (Für eine unter Diskreter-Logarithmus-Annahme sichere Hashfunktion für feste Länge kann man das Commitment-Schema aus der Vorlesung nehmen.)

Ausnahmen von der modularen Konstruktion, z.B. MDx, SHS, sollen hier nicht betrachtet werden (alles sehr ad-hoc).

C. Primitive

Anmerkung: Was Primitive sind (also Bausteine) und was „richtige" Protokolle, ist eigentlich ganz willkürlich.

C.1 Byzantinische Übereinstimmung (Broadcast-Protokolle)

Zuverlässige Verteilung von Nachrichten, so daß wirklich alle dieselbe bekommen.

Überblick: [Reis_87]

a) Einleitung:
   Erklärung des Problems; wieso BÜ als Protokoll und keinen physikalischen Broadcastkanal.
b) Informationstheoretisch sichere BÜ für $t < n/3$ (n = #Prozessoren, t = Maximalzahl fehlerhafter) und Beweis, daß es ohne Authentisierung für $t \geq n/3$ nicht geht [LaSP_82].
c) Vorstellung des kryptologisch sicheren BÜ-Protokolls für beliebiges t aus [DoSt_83].
d) Varianten des BÜ kurz skizzieren [Reis_87], insbes. andere Fehlermodelle (§1.2) und asynchrones Netzwerk (§3.1).

e) Effizienzkriterien kurz skizzieren [Reis_87, insbes §2.2 und §3.2].

C.2 Zero-Knowledge Beweise

Beweise, bei denen man nichts überflüssiges verrät. (Z.B. daß eine Zahl *n* wirklich aus zwei Primfaktoren besteht.)

Im wesentlichen nach [GoMW_91], und zwar vor allem

a) die Definition,

b) den Beweis für Graph-3-Colorability und

c) die Idee, warum das ZKP für alle Sprachen in NP ergibt.

Als Primitiv Commitments wählen. (Im Protokoll steht immer „encryption scheme“, aber die angegebene Definition ist sowieso eher die eines commitment schemes, und vgl. S. 712 oben; zudem sind commitments das „schwächere“ Primitiv. Dazu könnte man noch kurz „Commitment aus probabilistic encryption“ beweisen (skizziert auch in [BrCC_88, Kap. 6.2.3], aber da steht auch nicht mehr, d.h. man muß sich noch kurz selbst ausdenken, was man mit den Schlüsseln macht).)

Als kurzer Überblick ohne Beweise (aber mit viel mehr Sätzen als benötigt) ist [Feig_92] zu empfehlen, nur so zur Einordnung.

D. Allgemeine Multi-Party-Protokolle und Grundlagen

D.1 Münzwurf (incl. Unmöglichkeit unter bestimmten Vertrauensmodellen)

a) Zunächst an Definition und die Implementierung aus der Vorlesung Kryptographie erinnern, d.h. man weiß schon, daß Münzwurf mit komplexitätstheoretischer Sicherheit für 2 Parteien geht, aber mit einem Abbruchproblem.

b) Dann den Beweis aus [Clev_86], daß man dieses Abbruchproblem bei 2 Parteien immer hat.

c) Bißchen diskutieren, wie schlimm das Abbruchproblem ist (bei Bits schlimm, bei großen Zahlen nicht so, vgl. [PfWa_90, Abschnitt 2.3.2].

d) Falls weniger als die Hälfte von *n* Parteien angreifen, geht es aber wieder (und dann gleich informationstheoretisch sicher). Das ist u.a. eine Folge allgemeiner Multi-party-computation-Sätze, siehe D.3. Spezielle Konstruktionen nur für Münzwurf (die in D.3 wieder als Teile auftauchen) gehen direkt mit Verifiable Secret Sharing (VSS). Dazu von [CGMA_85] die Einleitung und die Anwendung aus Kap. 2.3. (Ein richtig gutes Protokoll für VSS ist D.2.) Zusammenhang zwischen VSS und Commitment (also Teil a und Teil c) rausarbeiten.

e) Falls Zeit bleibt, lieber nicht zuviel [CGMA_85], sondern für Vorsicht mit Münzwurfprotokollen sorgen: Unveröffentlichten Angriff von mir auf ein Münzwurfprotokoll von Ivan Damgård ähnlich dem aus der Vorlesung [Pfit?_91].

D.2 Verifiable Secret Sharing (VSS)

Verteilung von Stückchen von Geheimnissen, wenn die anderen dem Verteilenden nicht vertrauen.

a) Motivation aus [CGMA_85], vor allem Abschnitt 2.3. (Zum Teil auch in D.1, aber lieber nochmal Bezug herstellen.)

b) Dann das beste existierende Schema: informationstheoretisch sicher, und es reicht, daß weniger als die Hälfte angreifen [RaBe_89]. (Daraus nur VSS, nicht „incomplete networks“ und „multi-party computations“. Mehr Details in [PfWa_92, Waid_91].)

c) Sollte hier Zeit bleiben oder alles zu schwierig werden, noch das viel effizientere, komplexitätstheoretisch sichere Schema aus [Pede_92].

D.3 Multi-party computation mit ehrlicher Mehrheit

Ziemlich allgemeine Protokolle für sichere Berechnungen bei mehreren Parteien.

Vor allem nach [BeGW_88], aber ohne Theorem 2.

a) Ziel erklären, dazu z.B. auch das ursprüngliche Beispiel aus [Yao_82]. Sagen, was drunter fällt und was nicht. (Z.B. Münzwurf von vielen Parteien und Wahlprotokoll — ja; Münzwurf von zweien und zero-knowledge wegen ehrlicher Mehrheit nicht; Zahlungssysteme auch nicht, da keine ehrliche Mehrheit vorausgesetzt und nicht nur eine Funktionsberechnung. Aber sonst schon eine Menge Sachen.)

b) Konstruktion aus [BeGW_88], zumindest die, wo man noch kein Verifiable Secret Sharing (VSS) braucht und damit auch nicht die fehlerkorrigierenden Codes.

c) Beispiel wäre nett.

d) Ich denke, es reicht, dann einfach zu sagen, daß man jetzt das ganze trickreich mit VSS gemäß vorigem Vortrag kombiniert. Wenn man partout noch Lust hat, auf [RaBe_89, S.82-84] umsteigen.

D.4 2-party computation (also nicht mit ehrlicher Mehrheit)

Ziemlich allgemeine Protokolle für sichere Berechnungen bei zwei Parteien.

Wir wissen nach D.3, daß bei ehrlicher Mehrheit „alles“ informationstheoretisch geht (wenn man von Effizienz mal absieht). Auch wissen wir, daß wir ohne ehrliche Mehrheit ein Abbruchproblem haben (D.1). Je nach unseren Zielen ist das ok (z.B. beim Millionärsproblem wie in [Yao_82], in D.3 vermutlich vorgetragen). Jetzt ist die Frage, ob wir ohne ehrliche Mehrheit immer noch „alles“ können bis auf den Abbruch. Antwort: informationstheoretisch nicht, aber unter kryptologischen Annahmen.

a) Zuhörer an diese Zusammenhänge erinnern.

b) [ChKu1_89] dafür, daß 2 Leute informationstheoretisch kein UND mit Minimum-Knowledge berechnen können. (Und somit auch sonst nicht viel.)

c) [ChDG_88] mindestens so weit, daß 2 Leute unter kryptologischen Annahmen doch ein UND berechnet haben. Für Wichtigkeit des „UND“ z.B. das Rendezvous-Beispiel. Die Annahme sind Bit Commitments mit Zusatzeigenschaften.

E. Spezielle Protokolle

Warum nicht einfach die allgemeinen Sätze aus D nehmen? Entweder effizientere Varianten, oder weil die Modelle nicht ganz passen.

E.1 Wahlen

Vor allem nach [Chau_81].

Zusammenhang mit Multi-party-computation: Man setzt vor allem keine ehrliche Mehrheit voraus. Außerdem ist es eine sehr einfache Funktion, also kann man auf effiziente Protokolle

hoffen. Allerdings akzeptiert man (um Broadcastkommunikation zwischen allen Teilnehmern zu sparen) in den effizienten Varianten gewisse ehrliche Zentralen (ähnlich den Auszählern).

Da Chaum fast nichts definiert, die Definitionen in [CoFi_85, Kap. 2] betrachten. Gegenüberstellen, welche dieser Eigenschaften Chaum's Schema erreicht, und welche zusätzlichen es noch hat. Zum Beispiel so gliedern:

a) Was sind die Ziele und das Vertrauensmodell des Protokolls aus [Chau_81]?
b) Wie funktioniert es? (Wer „Sicherheit in Rechnernetzen“ gehört hat, kann das MIX-Kapitel sinnvoll verwenden.)
c) Alternative Ziele und Vertrauensmodelle?
d) Als Warnung bezüglich der Sicherheit eventuell noch die Ergebnisse aus [PfPf_90]. (Die Details des Angriffs führen wohl zu weit.)

E.2 Austausch von Vertragsexemplaren (Contract signing)

Hier wollen 2 Parteien einen Vertrag so unterschreiben und austauschen, daß entweder beide oder keiner am Schluß einen gültigen Vertrag haben.

Hier kann man das Abbruchproblem, vgl. D.1, sicher nicht vernachlässigen. Also fällt Vertragsaustausch nicht unter die Techniken aus D.4. (Man ahnt auch schon, daß es wirklich nicht perfekt geht, ähnlich wie der Münzwurf.) In diesem Vortrag wird vorgestellt, womit man sich behilft. (Es gibt übrigens Arbeiten, die solche Hilfslösungen wieder auf alle Protokolle aus D.4 verallgemeinern.)

Modelle und Protokolle aus [EvGL_85, BGMR_90].

Evtl. für „Geld gegen Ware“ noch die Idee aus [BüPf_90] skizzieren.

E.3 Poker

Im wesentlichen nach dem Ursprungsartikel [SRA_81]. Den „Unmöglichkeitsbeweis“ in Beziehung zu D.1 setzen, das Protokoll ein bißchen zu D.4.

Angriffe: [Cop1_86]. (Der darin dargestellte von Lipton 1979 war einer der Anlässe für die Entwicklung präziser Sicherheitsdefinitionen in der Kryptographie.)

Erweiterungen: Eine der Einleitungen von [Cre1_86, Crep_87] dafür, was man alles an Detaileigenschaften verlangen kann. Als Reparatur gegen den Lipton-Angriff aber höchstens [GoMi_82, Kap. 5] versuchen.

E.4 Zahlungssysteme

a) Einleitung: Schutzziele, Grundideen, Geräte aus [PWP_87] (in Kap. 5). Wieso sollten die Geräte Display und Tastatur besitzen?
b) Online-Zahlungssystem [Chau_89].
c) Offline-Zahlungssystem [ChFN_90]. Anmerkung über Observer.
d) Kurze Anmerkung zur derzeitigen Praxis (Dazu evtl. noch praktische Teile von [BBCM1_94].)

F. Sonstiges

F.1 Kryptographische Hierarchien

Bisher haben wir uns nicht sehr darum gekümmert, unter *welcher* kryptologischen Annahme etwas geht, sondern nur, ob überhaupt unter einer. Wenn man das auch noch betrachten will, ist es gut zu modularisieren. Beispielsweise wurden die Protokolle in D.3 aus Commitments konstruiert. Somit braucht man für konkrete kryptographische Annahmen nur noch zu zeigen, daß man daraus Commitments machen kann, und hat dann gleich ganz viele Protokolle. Umgekehrt fragt man sich, ob man aus jedem einzelnen Protokoll (z.B. aus Münzwurf) Bit Commitments machen kann, oder ob es möglich ist, daß unter manchen kryptologischen Annahmen Münzwurf möglich ist, aber Commitments nicht (so daß man den Münzwurf ganz anders konstruieren müßte als in D.1).

Der bekannteste Aufhänger für solche Modularisierungen ist die Einwegfunktion. Man kann von sehr vielen (allen?) Sachen, die nicht informationstheoretisch gehen, zeigen: Wenn es sie überhaupt gibt, gibt es auch Einwegfunktionen. Die Annahme „Existenz einer Einwegfunktion“ ist also fast immer notwendig, und man fragt sich, ob sie auch für fast alles hinreichend ist.

Für viele Sachen weiß man das tatsächlich, aber nicht für alle, und zwar insbesondere nicht für asymmetrische Konzelation und für informationstheoretisch geheimhaltende Commitments. (Für erstere gibt es sogar eine komplexitätstheoretische Einordnung, die zeigt, daß so ein Beweis nicht ganz einfach sein könnte.)

Aus diesem Gesamtrahmen sollen ein paar Resultate gezeigt werden. Insbesondere:

a) Was ist eine Einwegfunktion: Einleitung von [ImLL_89]?
b) Informationstheoretisch festlegende Commitments implizieren Einwegfunktionen [ImLu_89] (muß ich noch anschauen, wenn zu schwer, geht [Romp_90] mit Signatursystemen auf jeden Fall).
c) Einwegfuktionen implizieren Pseudozufallszahlengeneratoren (PRNG) [ImLL_89]. Definition von PRNG und Konstruktion nur skizzieren.
d) PRNG impliziert wieder Commitments [Naor_91]. Das kann man präzis darstellen.

Beachte: Insgesamt hat man jetzt nach C.2 und D.4 auch Zero-Knowledge-Beweise und 2-Parteien-Protokolle aus 1-weg-Funktionen.

F.2 Beweise mit Zufallsfunktions-Analogie (Random Oracle Hypothesis)

Nach [BeRo1_93].

Es gibt in der Kryptographie viele Techniken, die man nicht unter irgendwelchen sinnvollen Annahmen beweisen kann, aber trotzdem verwendet, weil sie effizienter sind als beweisbare Protokolle für den gleichen Zweck, z.B. das Verwenden von Hashfunktionen, um Signatursysteme gegen existentielle Fälschung sicher zu machen (vgl. Vorlesung).

Beweise mit Zufallsfunktions-Analogie sind ein Versuch, zu zeigen, daß hinter solchen Konstruktionen zumindest ein gewisser Sinn liegt: Man zeigt, daß die Konstruktion (im Mittel) sicher *wäre*, wenn man statt der Hash-Funktion eine völlig zufällige Funktion einsetzen würde, von der man an Funktionswerte nicht anders kommt, als daß ein „Orakel“ sie einem sagt.

Man beachte, daß wesentlich bekanntere Techniken für „Beweise“ kryptographischer Protokolle, etwa die sogenannte BAN-Logik, auf noch wesentlich unfundierteren Analogien beruhen, nur daß das dort nicht gesagt wird.

Aufgaben

Aufgaben 0

(Anwesenheitsübung in 2. Woche)

Überlegen

0.1 Wie viele Komponenten könnte ein Signatursystem haben und wie viele verschiedene Programme müssen wir für sie schreiben? (Und wie viele Hardwaregeräte würden wir wohl für sie bauen?)

0.2 Wenn jemand vom FTP-Server im Rechenzentrum ein Programm holt, um ihre Dateien zu verschlüsseln, wem muß sie da alles vertrauen? Manchen als *k*-aus-*n* o.ä., oder muß jeder einzelne vertrauenswürdig sein? (In anderen Worten: Ist es ein Serien- oder ein Parallelsystem bzgl. Vertrauen?) Wie kann man die Situation verbessern?

0.3 Wenn beim Secret Sharing der Direktor die Angestellten nicht persönlich zu sich ruft, sondern ihnen ihre Shares (Geheimnisteile) zuschickt, was muß er beachten? (Und welche Systemklassen braucht man?)

Rechnen u.ä.

0.4 Man zeige, daß die Formel $(K_i\phi \rightarrow \phi)$ allgemeingültig ist. Und was bedeutet sie?

Aufgaben 1

Reine Selbstkontrolle

1.1 Wo bzw. wodurch ersetzt treten die „Betroffenen“ in den Spezifikationen auf? Und im Vertrauensmodell?

1.2 Was sind die wichtigsten Vertrauensgrade in der Kryptologie?

1.3 Was unterscheidet Signatursysteme von Authentikationssystemen?

Überlegen

1.4 Man nehme an, man bekommt ein System aus drei Programmen angeboten: Am Anfang soll man mit einem davon eine Geheimzahl wählen („geheimer Schlüssel“), die man jemand zweitem mitteilt. Ab da kann man mit dem zweiten Programm zu einer Nachricht unter Eingabe des geheimen Schlüssels eine „Checkzahl“ erzeugen, die Signatur genannt wird. Mit dem dritten Programm kann man, wieder mit dem geheimen Schlüssel, zu einer Nachricht mit Checkzahl prüfen, ob sie zusammenpassen. Kann das ein Signatursystem sein? Was kann es sonst sein?

1.5 Wenn es nicht ein Direktor und 5 Angestellte sind, sondern ein Direktorat aus 5 Direktoren, denen der Tresor gehört, und die den Tresor auch öfter mal öffnen wollen, was ändert sich am Vertrauensmodell, und was könnte man tun? Braucht man überhaupt noch Secret Sharing? Gibt es eine sinnvolle Anwendung für Secret Sharing, auch wenn man die Geheimzahl mehr als einmal braucht? Kann man sie verallgemeinern auf „alle Anwendungen mit folgendem Vertrauensmodell“?

1.6 Wir haben mündlich den Münzwurf per e-mail durch „Münzwurf per Paketpost“ mit verschlossenen Holzkästchen betrachtet. Das kryptographische Analogon heißt Bit Commitment. Weshalb reicht nicht ein beliebiges Konzelationssystem? Würde Hinzunahme von Authentikation oder Signaturen helfen?

Rechnen u.ä.

1.7 Wissenslogik:

a) Zum Wetterbeispiel: Man drücke formal aus und zeige: In Welt 4 weiß Pierre, daß Hilde nicht weiß, daß es in Paris regnet. (Es lohnt sich also z.B., ihr darüber einen Brief zu schreiben.)

b) Was heißt die Formel $K_i\phi \to K_i\, K_i\phi$ anschaulich? Ist sie allgemeingültig?

1.8 Geheimhaltung mit Wahrscheinlichkeiten:

a) Man zeige, daß in der Interpretation mit Wahrscheinlichkeiten aus der Vorlesung das Wetter in Paris für H nicht nur in der Welt w_1, sondern absolut geheim ist.

b) Gibt es wohl noch andere sichere Verfahren für 3-aus-3 Secret Sharing außer dem angegebenen?

c) Cäsar-Chiffre: Wähle geheimen Buchstaben $\beta \in \{\mathrm{A}, \ldots, \mathrm{Z}\}$. Verschlüssele durch „Addition von β mod 26“ zu Buchstaben der Nachricht.

(Wie heißt das übrigens, einen Begriff wie „Addition“ einfach auf einer anderen Menge zu verwenden?)

Kann man mit dieser Chiffre eine drei Buchstaben lange Nachricht absolut geheim halten? (Die Antwort ist nicht einfach ja oder nein.)

Aufgaben 2

Reine Selbstkontrolle

2.1 Was war eine probabilistische Funktion?

2.2 Wenn eine Leitung als Komponente modelliert wird, wie viele Zugangspunkte hat sie? Und eine Leitung mit einem Abhörer?

2.3 Was waren ein Ablauf und ein Verhalten?

Überlegen

2.4 Wenn der Tresor beim Secret Sharing nur drei Versuche zuläßt, d.h. sich nach der dritten Eingabe einer falschen Geheimzahl endgültig verschließt, was für ein Problem kann sich ergeben? Wie sehen die Vertrauensmodelle genau aus, bei denen wir das Problem haben bzw. nicht haben? Wenn wir es haben: welche andere Systemklasse kann man sinnvoll einsetzen, um es zu lösen?

2.5 Ist es denkbar, daß man geheime e-mail-Nachrichten mit jemand austauschen kann, dessen e-mail Adresse man (woher auch immer) zuverlässig zu kennen glaubt, den man aber nicht persönlich kennt, wenn man sonst niemand vertraut (auch nicht k-aus-n u.ä.)?

2.6 Wenn man eine probabilistische Komponente wirklich implementieren will, wo können die Zufallszahlen herkommen?

Rechnen u.ä.

2.7 Wie könnte 4-aus-4 Secret Sharing für Geheimzahlen aus der Menge {1900, 1901, …, 1999} gehen?

2.8 Was tut das System mit folgender echt probabilistischer Übergangsfunktion anschaulich? (Man betrachte am besten zuerst mögliche Abläufe.):
$I = \{0,1\} \times \{0,1\}$, $O = \{0,1\}$;
für alle n:

$$f_n((i_1, \ldots, i_n), (o_1, \ldots, o_{n-1}))(o_n) = \begin{cases} 1 & \text{falls } (n = 1 \text{ oder } i_{n-1,2} = 0) \text{ und } o_n = 0; \\ 0{,}9 & \text{falls } i_{n-1,2} = 1 \text{ und } o_n = i_{n-1,1}; \\ 0{,}1 & \text{falls } i_{n-1,2} = 1 \text{ und } o_n \neq i_{n-1,1}; \\ 0 & \text{sonst.} \end{cases}$$

Hierbei bezeichnen $i_{x,1}$ und $i_{x,2}$ die erste und zweite Komponente von i_x.

2.9 Secret Sharing mit „allgemeiner Zugangsstruktur“: Beim Secret Sharing müssen nicht immer alle Leute symmetrische Rechte haben (so daß beliebige k aus n die Geheimzahl rekonstruieren können), sondern irgendwelche Teilmengen verschiedener Größe.

a) Man gebe ein sicheres Verfahren für folgende Situation an: Es gibt 6 Leute, $\{A, \ldots, F\}$. Die Menge $\{A, B, C\}$ soll die Geheimzahl rekonstruieren können, ebenso $\{A, C, D\}$, und E und F zusammen sollen jeweils als Vertreter für A einspringen können.

b) Ist M die Menge aller betrachteten Leute, so nennt man die Menge Z derjenigen Teilmengen, die die Geheimzahl rekonstruieren dürfen, eine *Zugangsstruktur*. Ist jede Menge $Z \subseteq P(M)$ eine sinnvolle Zugangsstruktur, oder was ist die allgemeinste Bedingung für sinnvolle Z?

2.10 Man überlege, noch besser beweise, wieso beim Secret Sharing jedes einzelne Share mindestens genauso lang sein muß wie die Geheimzahl. (Genauer: Wenn die Geheimzahl aus einer Menge mit n Elementen stammt, muß auch jedes Share mindestens n Werte annehmen können.)

Aufgaben 3

Überlegen

3.1 Für Kommunikationsleitungen gibt es z.B. Codes, die zugleich 1-fehler-korrigierend und 2-fehler-erkennend sind: Was sind hier die Ziele und das Vertrauensmodell? Wie realistisch ist es?

3.2 Zwei Leute tauschen per Diskette einen langen Zufallsstring z aus. Jedesmal, wenn sie eine Nachricht N verschicken, wollen sie ein neues Stück von z mitschicken, als Zeichen, daß die Nachricht von einem Besitzer von z kommt. Damit nicht auf der Leitung die Nachricht N gegen eine andere vertauscht wird, hängen sie z nicht einfach hinten dran, sondern schicken statt dessen $N \oplus z$. Was ist davon zu halten?

a) Was für einen Systemtyp wollen diese Leute?

b) Ist ihr System sicher?

c) Sofern man Angriffe findet, sind diese wohl praktisch relevant?

3.3 a) Kann man beim Shamir-Schema für Secret Sharing leicht einen neuen Teilnehmer hinzunehmen? Das heißt, nachdem man das Geheimnis schon verteilt hat, möchte man eine k-aus-(n+1)-Verteilung statt k-aus-n.

b) Könnte man statt dessen k erhöhen? Oder einen Teilnehmer rauswerfen, also n senken? Oder k senken? (Nicht alles geht, jedenfalls nicht einfach.)

Rechnen

3.4 a) Was besagt die folgende temporallogische Formel anschaulich?

$$\Box \big(\mathit{eingabe}(z, \text{„}\mathit{stop}\text{“}) \rightarrow \Box\, \mathit{ausgabe}(z, \mathit{none}) \big)$$

Wie lautet sie in normaler Prädikatenlogik (über Folgen)?

b) Und diese?

$$\big(\mathit{eingabe}(z, \text{„}\mathit{arbeite}\text{“}) \wedge \bigcirc\, \mathit{eingabe}(z^*, \text{„}\mathit{arbeite\_mit}\text{“}) \big)$$
$$\rightarrow \big(\Diamond\, \exists\, N \neq \mathit{none}\colon \mathit{ausgabe}(z, N) \wedge \mathit{ausgabe}(z^*, N) \big)$$

c) Gibt es ein System, das (bei beliebigen Benutzern) beide Formeln erfüllt? Und wenn man ganz außen um die zweite Formel noch ein $\Box$ getan hätte?

3.5 Man spezifiziere (wie in Kap. 2.2.2 angedeutet) ein perfektes, reaktives Münzwurfsystem, mit dem man beliebig oft Münzen werfen kann, durch Angabe eines zentralen, vertrauenswürdigen Vergleichssystems. (Ein erweiterter endlicher Automat bietet sich als Spezifikationstechnik an.)

3.6 Man führe ein Beispiel für 2-aus-3 Secret Sharing im Körper $\mathbb{Z}_5$ durch. Es gebe insgesamt eine 4-Bit-Zahl, die man so aufteilen möchte, und die dazu in zwei Blöcke zerlegt wird.

Ist eigentlich das Secret Sharing, wenn es so als mehrfach benutzbares reaktives System aufgefaßt wird, sicher, falls es Kontakte zwischen den zu schützenden Benutzern und den Angreifern gibt (wie in Kap. 3.3.2)? Wie würden sich solche Angriffe hier überhaupt äußern?

Aufgaben 4

Überlegen

4.1 Wenn man einen Authentikationscode mit der Fehlerwahrscheinlichkeit $2^{-\sigma}$ hat, man hätte aber lieber die Fehlerwahrscheinlichkeit $2^{-\tau}$ mit $\tau > \sigma$, was kann man tun? Ist das effizient?

4.2 Wenn man einen Authentikationscode für x Authentikationen initialisiert hat und merkt, daß einem bald der Schlüssel ausgeht, kann man die noch vorhandenen Authentikationsmöglichkeiten nutzen, um einen neuen Schlüssel einfacher auszutauschen?

4.3 Wenn man einen Authentikationscode für mittellange Nachrichten benutzt, und eine Nachricht ist doch länger als erwartet, was kann man tun?

4.4 Bei den Authentikationscodes müssen Schlüssel geheim ausgetauscht werden, obwohl nachher die Nachrichten nicht geheim sind. Bei Signatursystemen hingegen ist der ausgetauschte Schlüssel öffentlich. Könnte man auch für Authentikationssysteme mit einem integren Kanal in der Initialisierung auskommen („asymmetrische Authentikation“)? Kann das informationstheoretisch sicher sein, d.h. gegen Angreifer, die beliebig viel rechnen können?

Zur Vereinfachung betrachte man (zunächst oder nur) nicht interaktive Schemata.

4.5 Kann es ein absolut sicheres Commitment Schema geben?

Rechnen

4.6 Bei den in der Vorlesung gezeigten Authentikationssystemen für mehr als eine Nachricht hatte der Algorithmus *auth* nicht nur *key* und *m*, sondern auch *Zähler* als Eingabe. Hier folgt ein Code, bei dem diese Eingabe nicht benötigt wird:

Beim Nachrichtenraum $\{0, 1\}^n$ und Sicherheitsparameter σ arbeitet man wieder in einem Körper K mit $|K| \geq \max\{2^n, 2^\sigma\}$.

gen: Bei Eingabe x, σ, wobei x die Anzahl der zu authentisierenden Nachrichten ist: Wähle $a_0, \ldots, a_x \in_R K$,

$$key = (a_0, \ldots, a_x)$$

$$auth(key, m) := a_0 + a_1 m + a_2 m^2 + \ldots + a_x m^x.$$

a) Man zeige, daß dieser Authentikationscode sicher ist, wenn man wirklich nur x Nachrichten damit authentisiert.

(Die Funktionenklasse ist sogar „strongly universal$_{(x+1)}$“. Was heißt das wohl?)

b) Wie gut ist die Effizienz (v.a. im Vergleich mit Codes aus der Vorlesung)?

c) Braucht die Senderkomponente jetzt noch die Variable *Zähler*?

d) Wie nützlich könnte die Eigenschaft, daß *auth*(*key*, *m*) keinen *Zähler* als Eingabe hat, in der Praxis sein?

4.7 Man denke sich ein Verfahren zur Replayvermeidung (oder gleich Timeliness) aus und schreibe es etwa so präzis auf wie die Komponenten in Kap. 4.2.4.

4.8 (Schreibweisen für mehrere probabilistische Algorithmen): Man betrachte folgendes Würfelspiel: *A* würfelt. *B* muß versuchen, dieselbe Zahl zu würfeln. *B* darf aber dabei zwischen einer Münze und einem Würfel wählen.

a) Wie kann man die Gewinnwahrscheinlichkeit für dieses Spiels hinschreiben?

b) Man gebe eine gute Strategie für *B* an.

c) Man berechne dafür die Gewinnwahrscheinlichkeit.

Aufgaben 5

Überlegen

5.1 Ergibt die Operation

$$encrypt(key, m) := m + 3key \bmod n,$$

wobei *key* zufällig modulo *n* gewählt wird, ein perfekt sicheres Konzelationssystem?

Rechnen

5.2 Nehmen wir an, die Formel für die Laufzeit des derzeit besten Faktorisierungsalgorithmus sei genau $C \cdot L(n)$ für eine Konstante C (d.h. ohne zusätzliche Konstanten im Exponenten). Wieviel mal solange brauchen dann die Leute, die vor kurzem eine Zahl mit 130 Dezimalstellen faktorisieren konnten, um statt dessen eine mit 140 Dezimalstellen zu faktorisieren?

Wenn sie immer nur gleichviel Rechenzeit auf ihren Rechnern bekommen, aber jedes Jahr doppelt so schnelle Rechner, in wie langer Zeit sind sie soweit?

5.3 Wenn wir uns nur auf die Faktorisierungsannahme aus der Vorlesung verlassen wollen, können wir dann auch Verfahren benutzen, in denen Zahlen $n = pqr$ mit drei gleich langen Primfaktoren p, q und r veröffentlicht werden, wo aber p, q, und r geheim bleiben müssen?

Falls ja, und falls wir in der Praxis $n = pq$ mit 660 bits nehmen, wie groß würden wir $n = pqr$ wählen?

5.4 Die Eulersche ϕ-Funktion ist definiert als

$$\phi(n) := |\{a \in \{1,\ldots,n-1\} \mid \mathrm{ggT}(a, n) = 1\}|.$$

für alle natürlichen Zahlen.

a) Was ist $\phi(n)$ für $n = pq$ mit p, q prim?

b) Man formuliere eine Annahme, daß das Berechnen von $\phi(n)$ schwer ist.

c) Man zeige, daß diese Annahme zur Faktorisierungsannahme äquivalent ist. (Der Hauptteil ist, zu zeigen, daß man aus n und $\phi(n)$ die Faktoren p und q relativ leicht bestimmen kann.)

Aufgaben 6

Überlegen

6.1 Wenn man $(\mathbb{Z}_n, +)$ durch einen Kreis veranschaulichen kann, wodurch kann man ein Produkt aus zwei zyklischen Gruppen veranschaulichen? (Also z.B. $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$.)

6.2 Nehmen wir an, wir haben eine Familie von *additiv* geschriebenen Gruppen.

a) Wie schreibt man dann die Funktion, die dem diskreten Logarithmus entspricht?

b) Ist der „diskrete Logarithmus“ in den Gruppen $(\mathbb{Z}_n, +)$ schwierig?

6.3 Wenn uns jemand eine merkwürdige Sorte von Gruppen vorschlägt (z.B. Gruppen von Matrizen oder „jacobians of hyperelliptic curves” — die gibt’s wirklich), darin eine diskrete-Logarithmus-Annahme macht und dann Kryptosysteme von „normalen“ Gruppen auf diese überträgt, was muß man beachten?

a) Bezüglich Sicherheit?

b) Was für effiziente Algorithmen sollte man für diese Gruppen wenigstens kennen, damit man mit ihnen sinnvolle Systeme bauen kann?

Rechnen

6.4 a) Man berechne $9^{25} \bmod 13$.

b) Gibt es eine bessere Additionskette für den Exponenten $(10011101110011)_2$ als die aus square-and-multiply? (Hinweis: Man bleibe am besten in der Binärdarstellung.)

6.5 Man bestimme eine Quadratwurzel von 46 mod 105, indem man sie bezüglich der Primfaktoren rät und mit dem Chinesischen Restsatz zusammensetzt.

6.6 Man berechne $392^{1443} \bmod 13$. (Hinweis: Es geht deutlich einfacher als mit square-and-multiply!)

6.7 Man suche (durch Probieren) einen Generator g von $\mathbb{Z}_{11}^*$ und stelle alle Gruppenelemente als Potenzen von g dar.

Aufgaben 7

Überlegen

7.1 Welcher Teilnehmer am Münzwurf in Kap. 4.2 ist informationstheoretisch sicher und welcher nicht (und wieso)?

7.2 Alice und Bob würfeln wie in Kap. 1.2 um die Waschmaschine, aber mit dem Verfahren aus Kapitel 6.2. Alice sei Teilnehmer 1. Wenn Bob ihr einmal glaubt, daß ihr Rechner kaputtgegangen ist, aber jetzt repariert wurde, und dann nochmal glaubt, daß er wieder kaputt ist, wie hoch kann dann Alice die Wahrscheinlichkeit machen, daß sie die Waschmaschine bekommt?

7.3 Wenn die Senderkomponente im Commitment aus Kapitel 6.1 nicht prüfen würde, daß die Parameter p, q, g, h korrekt sind, wie könnte dann der Empfänger mogeln?

Und was wäre, wenn statt dessen die Senderkomponente diese Parameter wählen dürfte?

Rechnen

7.4 a) Man berechne $\sqrt[5]{8} \bmod 17$.

b) Wie sieht es mit $\sqrt[5]{8} \bmod 11$ aus?

c) Und $\sqrt[7]{19} \bmod 33$?

7.5 Als Fortsetzung von Aufgabe 6.7: Was sind die Untergruppen von $\mathbb{Z}_{11}^*$? Und ihre Generatoren?

7.6 Man spiele ein Beispiel des Commitment-Schemas aus Kapitel 6.1 durch.

(Wenn es klein genug ist, kann man auch schauen, wie das Commitment beim Brechen der diskreten-Logarithmus-Annahme anders zu öffnen wäre.)

Aufgaben 8

Überlegen

8.1 Das RSA-Basissignatursystem geht einfach so, daß alles wie in der RSA-Annahme initialisiert wird und dann die Signiererkomponente aus jedem Nachrichtenblock $m \in \mathbb{Z}_n$ als Signatur die e-te Wurzel zieht. Als Test potenziert man die Signatur mit e.

Wie kann man hier fälschen (existential forgery ohne vorherigen chosen-message Angriff)?

8.2 Die strongly universal$_2$ Funktionen, die wir als Authentikationscodes verwendet haben, wurden ursprünglich auch als Hashfunktionen für Datenspeicherung mit Hashtabellen erfunden. Würden sie auch als kryptologische Hashfunktionen in Verbindung mit einem Signatursystem wie ElGamal taugen? Warum? (Beweisidee bzw. konkreten Angriff für Funktionen $hash((a, b), m) = a + b \bullet m|_{\sigma}$.)

Rechnen

8.3 Man rechne ein kleines Beispiel für eine Unterschrift mit dem ElGamal-Basissystem und ihren Test.

8.4 a) Die DSS-Testgleichung ist

$$r = (g^{hash(m)\bullet s^{-1}} h^{r\bullet s^{-1}} \bmod p) \bmod q,$$

wobei die Rechnungen im Exponenten wie üblich modulo q sind. Wie geht ein Signieralgorithmus dazu?

Hinweis: Das etwas unmotivierte „mod q“ am Ende kann man als eine Funktion $hash*$ betrachten, die nur ganz am Schluß die Zahlen mod p (z.B. 512 bits) verkürzt (z.B. auf 160 bits).

b) Wie lang ist eine Signatur?

c) Kann man beim Signieren wieder etwas vorwegberechnen, bevor man die Nachricht kennt?

d) Kann man beim Testen wieder Exponentiationen zusammenfassen?

8.5 Man berechne $7^{2^{500}} \bmod 33$.

Literatur

Beachte: In diesem Skript wurde nicht alle zugrundeliegende Literatur zitiert, sondern nur Artikel, in denen man mehr zu knappen Anmerkungen im Text oder im Anhang finden kann. Für vollständigere Bibliographien siehe die eingangs genannten Bücher.

BachE_90 Eric Bach: Intractable Problems in Number Theory; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 77-93.

BBCM1_94 Jean-Paul Boly, Antoon Bosselaers, Ronald Cramer, Stig Mjølsnes, Frank Muller, Torben Pedersen, Birgit Pfitzmann, Peter de Rooij, Berry Schoenmakers, Luc Vallèe, Michael Waidner: The ESPRIT Project CAFE— High Security Digital Payment Systems —; ESORICS 94 (Third European Symposium on Research in Computer Security), Brighton, LNCS 875, Springer-Verlag, Berlin 1994, 217-230.

BeGW_88 Michael Ben-Or, Shafi Goldwasser, Avi Wigderson: Completeness theorems for non-cryptographic fault-tolerant distributed computation; 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 1-10.

Bena_87 Josh Cohen Benaloh: Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract); Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 251-260.

BeRo1_93 Mihir Bellare, Phillip Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols; 1st ACM Conference on Computer and Communications Security, Proceedings, Fairfax, November 1993, acm Press, New York 1993, 62-73.

BGMR_90 Michael Ben-Or, Oded Goldreich, Silvio Micali, Ronald L. Rivest: A Fair Protocol for Signing Contracts; IEEE Transactions on Information Theory 36/1 (1990) 40-46.

BrCC_88 Gilles Brassard, David Chaum, Claude Crépeau: Minimum Disclosure Proofs of Knowledge; Journal of Computer and System Sciences 37 (1988) 156-189.

BrDL_93 Jørgen Brandt, Ivan Damgård, Peter Landrock: Speeding up Prime Number Generation; Asiacrypt '91, LNCS 739, Springer-Verlag, Berlin 1993, 440-449.

BüPf_90 Holger Bürk, Andreas Pfitzmann: Value Exchange Systems Enabling Security and Unobservability; Computers & Security 9/8 (1990) 715-721.

CGMA_85 Benny Chor, Shafi Goldwasser, Silvio Micali, Baruch Awerbuch: Verifiable secret sharing and achieving simultaneity in the presence of faults; 26th Symposium on Foundations of Computer Science (FOCS) 1985, IEEE Computer Society, 1985, 383-395.

Chau_81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

Chau_89 David Chaum: Privacy Protected Payments – Unconditional Payer and/or Payee Untraceability; SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20.10.1987, North-Holland, Amsterdam 1989, 69-93.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChFN_90 David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.

ChKu1_89 Benny Chor, Eyal Kushilevitz: A Zero - One Law for Boolean Privacy; 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 62-72.

Clev_86 Richard Cleve: Limits on the Security of Coin Flips When Half the Processors are Faulty; 18th Symposium on Theory of Computing (STOC) 1986, ACM, New York 1986, 364-369.

CoFi_85 Josh D. Cohen, Michael J. Fischer: A robust and verifiable cryptographically secure election scheme; 26th Symposium on Foundations of Computer Science (FOCS) 1985, IEEE Computer Society, 1985, 372-382.

Cohe1_93 Henri Cohen: A Course in Computational Algebraic Number Theory; Graduate Texts in Mathematics 138, Springer-Verlag, Berlin 1993.

Cop1_86 Don Coppersmith: Cheating at Mental Poker; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 104-107.

Cre1_86 Claude Crepeau: A Secure Poker Protocol that Minimizes the Effect of Player Coalitions; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 73-86.

Crep_87 Claude Crépeau: A zero-knowledge Poker protocol that achieves confidentiality of the players' strategy or How to achieve an electronic Poker face; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 239-247.

Damg_88 Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.

DaPr_89 Donald W. Davies, Wyn L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer; (2nd ed.) John Wiley & Sons, New York 1989.

DiLe_94 Brandon Dixon, Arjen K. Lenstra: Factoring Integers using SIMD Sieves; Eurocrypt '93, Lofthus, Norwegen, Mai 1993, Proceedings, LNCS 765, Springer-Verlag, Berlin 1994, 28-39.

DoSt_83 Danny Dolev, H. Raymond Strong: Authenticated Algorithms for Byzantine Agreement; SIAM J. Comput. 12/4 (1983) 656-666.

EvGL_85 Shimon Even, Oded Goldreich, Abraham Lempel: A Randomized Protocol for Signing Contracts; Communications of the ACM 28/6 (1985) 637-647.

Feig_92 Joan Feigenbaum: Overview of Interactive Proof Systems and Zero-Knowledge; Gustavus J. Simmons: Contemporary Cryptology – The Science of Information Integrity; IEEE Press, Hoes Lane 1992, 423-439.

FiSh_87 Amos Fiat, Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 186-194.

GoMi_82 Shafi Goldwasser, Silvio Micali: Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information; 14th Symposium on Theory of Computing (STOC) 1982, ACM, New York 1982, 365-377.

GoMR_88 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

GoMW_91 Oded Goldreich, Silvio Micali, Avi Wigderson: Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems; Journal of the ACM 38/1 (1991) 691-729.

Gord_85 John Gordon: Strong Primes are Easy to Find; Eurocrypt '84, LNCS 209, Springer-Verlag, Berlin 1985, 216-223.

Gord1_93 Daniel M. Gordon: Discrete logarithms in GF(p) using the number field sieve; SIAM Journal on Discrete Mathematics 6/1 (1993) 124-138.

Halp_87 Joseph Y. Halpern: Using Reasoning about Knowledge to Analyze Distributed Systems; Ann. Rev. Comput. Sci. 1987, 37-68.

HePe_93 Eugène van Heyst, Torben P. Pedersen: How to make efficient Fail-stop signatures; Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 366-377.

ImLL_89 Russell Impagliazzo, Leonid A. Levin, Michael Luby: Pseudo-random Generation from One-way Functions; 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 12-24.

ImLu_89 Russell Impagliazzo, Michael Luby: One-Way Functions are Essential for Complexity Based Cryptography; 30th Annual Symposium on Foundations of Computer Science (FOCS) 1989, IEEE Computer Society, 1989, 230-235.

Knut_81 Donald E. Knuth: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (2nd ed.); Addison-Wesley, Reading 1981.

LaMa_91 Xuejia Lai, James L. Massey: A Proposal for a New Block Encryption Standard; Eurocrypt '90, LNCS 473, Springer-Verlag, Berlin 1991, 389-404.

LaMa_93 Xuejia Lai, James L. Massey: Hash functions based on block ciphers; Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 55-70.

LaOd_91 Brian A. LaMacchia, Andrew M. Odlyzko: Computation of Discrete Logarithms in Prime Fields; Designs, Codes and Cryptography 1/1 (1991) 47-62.

LaSP_82 Leslie Lamport, Robert Shostak, Marshall Pease: The Byzantine Generals Problem; ACM Transactions on Programming Languages and Systems 4/3 (1982) 382-401.

LeLe_90 Arjen K. Lenstra, H. W. Lenstra, Jr: Algorithms in Number Theory; in: J. van Leeuwen (ed.): Handbook of Theoretical Computer Science; Elsevier Science Publishers B. V., 1990, 673-715.

LeLe_93 A. K. Lenstra, H. W. Lenstra: The development of the number field sieve; Lecture Notes in Mathematics 1554, Springer Verlag, Berlin 1993.

LeMa1_90 Arjen K. Lenstra, Mark S. Manasse: Factoring by electronic mail; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 355-371.

MaPn1_91 Zohar Manna, Amir Pnueli: The Temporal Logic of Reactive and Concurrent Systems: Specification; Springer-Verlag, New York 1991.

Naor_91 Moni Naor: Bit Commitment Using Pseudorandomness; Journal of Cryptology 4/2 (1991) 151-158.

Pede_92 Torben P. Pedersen: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing; Crypto '91, LNCS 576, Springer Verlag, Berlin 1992, 129-140.

Pfit?_91 Birgit Pfitzmann: Problems with CDG-Multi-party Computations; Notes for Ivan Damgård, Universität Karlsruhe, 4.3.1991.

PfPf_90 Birgit Pfitzmann, Andreas Pfitzmann: How to Break the Direct RSA-Implementation of MIXes; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 373-381.

PfWa_90 Birgit Pfitzmann, Michael Waidner: Formal Aspects of Fail-stop Signatures; Interner Bericht 22/90 der Fakultät für Informatik, Universität Karlsruhe, Dezember 1990.

PfWa_92 Birgit Pfitzmann, Michael Waidner: Unconditional Byzantine Agreement for any Number of Faulty Processors; STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, LNCS 577, Springer-Verlag, Heidelberg 1992, 339-350.

PfWa_94 Birgit Pfitzmann, Michael Waidner: A General Framework for Formal Notions of "Secure" System; Hildesheimer Informatik-Berichte 11/94, ISSN 0941-3014, Institut für Informatik, Universität Hildesheim, April 1994.

PfWa2_91 Birgit Pfitzmann, Michael Waidner: Fail-stop Signatures and their Application; Securicom 91; 9th Worldwide Congress on Computer and Communications Security and Protection, Paris, 19.-22. March 1991, 145-160.

PoHe_78 Stephen C. Pohlig, Martin E. Hellman: An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance; IEEE Transactions on Information Theory 10/1 (1978) 106-110.

Poll_74 J. M. Pollard: Theorems on factorization and primality testing; Proceedings of the Cambridge Philosophical Society, Band 76, 1974, 521-528.

Pome_84 Carl Pomerance: Analysis and comparison of some integer factoring algorithms; Computational Methods in Number Theory Band 1; H. W. Lenstra jr., R. Tijdeman (eds.); Mathematical Centre Tracts 154 CWI Amsterdam (ehemals Mathematisch Centrum), Amsterdam 1984, 89-139.

Pome_85 Carl Pomerance: The Quadratic Sieve Factoring Algorithm; Eurocrypt '84, LNCS 209, Springer-Verlag, Berlin 1985, 169-182.

Pome1_90 Carl Pomerance: Factoring; Carl Pomerance (ed.): Cryptology and Computational Number Theory; Proceedings of Symposia in Applied Mathematics, Volume 42, American Mathematical Society, Providence 1990, 27-47.

PrGV_94 Bart Preneel, René Govaerts, Joos Vandewalle: Hash functions based on block ciphers: a synthetic approach; Crypto '93, Springer-Verlag, Berlin 1994, 368-378.

PWP_87 Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Computer und Recht 3/10,11,12 (1987) 712-717, 796-803, 898-904; Überarbeitung und Erweiterung erschien in zwei Teilen in Datenschutz und Datensicherung DuD 14/5-6 (1990) 243-253, 305-315.

RaBe_89 Tal Rabin, Michael Ben-Or: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority; 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 73-85.

Rabi_79 Michael O. Rabin: Digitalized Signatures and Public-Key Functions as Intractable as Factorization; Massachusetts Institute of Technology, Laboratory for Computer Science, MIT/LCS /TR-212, January 1979.

Reis_87 Rüdiger Reischuk: Konsistenz und Fehlertoleranz in Verteilten Systemen - Das Problem der Byzantinischen Generäle; 17. GI Jahrestagung, IFB 156, Springer-Verlag, Berlin 1987, 65-81.

RIPE1_93 RIPE Consortium: RIPE integrity primitives: Final report of RACE 1040; Centrum voor Wiskunde en Informatica, Computer Science/Departement of Algorithmics and Architecture, Report CS-R9324 und -R9325, April 1993.

Rive2_91 Ronald L. Rivest: The MD4 Message Digest Algorithm; Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 303-311.

Romp_90 John Rompel: One-Way Functions are Necessary and Sufficient for Secure Signatures; Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC), May 14-16, 1990, Baltimore-Maryland, ACM Press, 387-394.

RSA_78 R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, nachgedruckt: 26/1 (1983) 96-99.

Schu_94 Matthias Schunter: Spezifikation von Geheimhaltungseigenschaften für reaktive kryptologische Systeme; Diplomarbeit am Institut für Informatik, Universität Hildesheim, Januar 1994.

Sham_79 Adi Shamir: How to Share a Secret; Communications of the ACM 22/11 (1979) 612-613.

SRA_81 Adi Shamir, Ronald L. Rivest, Leonard M. Adleman: Mental Poker; The Mathematical Gardner; David Klarner (ed.), Boston, 1981, 37-43.

Waid_91 Michael Waidner: Byzantinische Verteilung ohne kryptographische Annahmen trotz beliebig vieler Fehler; Universität Karlsruhe, Fakultät für Informatik, Dissertation, 24. Oktober 1991.

Yao_82 Andrew C. Yao: Protocols for Secure Computations; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 160-164.

ZhSe1_93 Yuliang Zheng, Jennifer Seberry: Practical Approaches to Attaining Security against Adaptively Chosen Ciphertext Attacks; Crypto '92, LNCS 740, Springer Verlag, Berlin 1993, 292-304.