Schlüsselvereinbarung

Seminar „Kryptografische Protokolle“
an der Humboldt Universität zu Berlin.

Stephan Verbücheln, Daniel Schliebner.

Herausgabe: 4. März 2009

Inhaltsverzeichnis

1 Einführung

1.1 Einleitung und Motivation

Thema dieser Arbeit ist die Einführung in Schlüsselvereinbarungsschemata (engl. „key agreements schemes“, kurz **KAS**). Um zu klären, was genau man unter einem solchen Schema oder Protokoll zur Schlüsselvereinbarung versteht, wollen wir uns überlegen, in welchen realen Szenarien es von Vorteil sein kann, einen Schlüssel *sicher austauschen zu können.* Insbesondere werden wir sehen, wie man den unscharfen Begriff *sicher* hinreichend gut konkretisieren kann.
Asymmetrische Verschlüsselungsverfahren (z. B. das RSA-Verfahren) benötigen bekannter Weise keine Schlüsselvereinbarungstechniken, da mithilfe öffentlicher, d. h. für jeden (also auch Angreifer) zugänglicher, Schlüssel Nachrichten versendet werden können. Es lässt sich also berechtiger Weise die Frage stellen, wozu eigentlich KAS benötigt werden. Die Motivation solcher Verfahren ist einfach: Public-Key Verfahren sind i. a. sehr aufwendig in der Ver- und Entschlüsselung - insbesondere bei längeren Geheimtexten (die Laufzeit ist etwa proportional zur Länge eines Geheimtextes). So ist z. B. das RSA-verfahren ca. 1000-fach langsamer bei der Encodierung langer Geheimtexte, als symmetrische Verfahren (z. B. AES). Dies hat zur Folge, dass in der Praxis oftmals symmetrische Verfahren statt asymmetrischer verwendet werden. *Die Motivation für KAS liegt also insbesondere in den für die Praxis relevanten symmetrischen Verschlüsselungsverfahren, welche einen Schlüsselaustausch notwendig machen.*

1.2 Ansätze

Formal befinden wir uns folglich innerhalb des Szenarios, dass sich zwei Personen U und V (diese Bezeichungen werden im folgenden stets verwendet) eine Nachricht verschlüsselt senden wollen. Zur Ver- und Entschlüsselung verwenden dabei U und V denselben Schlüssel K (solche Verfahren werden auch *symmetrisch* genannt).
Das Problem, was sich nun unmittelbar stellt, besteht darin, wie die Vereinbarung von K von statten gehen kann, ohne dass potentielle Angreifer W Wissen über diesen Schlüssel K erlangen können. Welche Sicherheitseigenschaften dabei zu erfüllen sind, werden wir später noch genauer betrachten.
Man unterscheidet i. a. zwischen folgenden Varianten, wie sich Schlüssel sicher vereinbaren lassen:

Definition 1 ***Schlüsselbereitsstellung*** *(engl. „key distribution“) beschreibt den Ansatz, dass K von einer Zertifizierungsstelle (engl. „trusted authority“, kurz **TA**) herausgegeben und über einen sicheren Kanal an U und V übermittelt wird.*

Definition 2 ***Schlüsselvereinbarung*** *(engl. „key agreement“) beschreibt das Problem, wie zwei Personen U und V über einen abhörbaren und manipulierbaren Kanal einen Schlüssel K vereinbaren können, ohne dabei Dritte (etwa TA's) wesentlich zu Hilfe ziehen zu müssen.*

Im Szenario der Schlüsselvereinbarung konstruieren sich U und V einen Schlüssel K, indem sie sogenannte **Schlüsselvereinbarungsschemata** (engl. „key agreements schemes“, kurz **KAS**) verwenden. KAS können ebenfalls kategorisiert werden; man unterscheidet:

1. Private-Key-Schemata (z. B. EKE2) und

2. Public-Key-Schemata (z. B. Diffie-Hellman-KAS).

In Ersterem wird vorausgesetzt, dass jeder der Beteiligten U und V einen privaten Schlüssel besitzt, mithilfe dessen sie wiederum den Schlüssel K verschlüsselt übermitteln können. Mit den Public-Key-Schemata wollen wir uns in dieser Arbeit auseinander setzen.

1.3 Probleme/Angriffe - der Begriff der Sicherheit

Wie bereits angekündigt, wollen wir uns vor konkreten KAS noch überlegen, wogegen ein solches Schema oder Protokoll sicher sein sollte. Hierzu kann man sich überlegen, dass prinzipiell folgende Angriffsszenarien denkbar sind:

- Veränderung von abgehörten Informationen;
- Speicherung von abgehörten Informationen;
- Versuche, sich als berechtigter Teilnehmer des Netzwerkes auszugeben.

Sind konkret U, V zwei Parteien des KAS. Dann könnte der Sinn und Zweck der soeben genannten Angriffe sein, dass ein Angreifer W versucht

1. U, V abgelaufene Schlüssel akzeptieren zu lassen;
2. U, V in den Glauben zu versetzen, sie hätten einen Schlüssel ausgetauscht, obwohl sie es nicht haben;
3. andere Informationen aus dem Austausch zu ziehen.

Zuletzt wollen wir noch eine weitere Klasse von KAS anschauen:

Definition 3 ***Authentificated KAS*** *sind Schemata, welche die Parteien U, V zusätzlich auch sicher identifizieren können. Dies geschieht i. a. mithilfe digitaler Signaturen (mehr dazu später).*

Diese Protolle werden im Verlauf dieser Arbeit interessant, wenn es darum geht, wie U und V sicherstellen können, dass vereinbarte Informationen in der Tat von der anderen Partei stammen und nicht von eventuellen Angreifern manipuliert wurden.

Man kann darüber hinaus im Wesentlichen zwei Klassen von Angreifern unterscheiden:

Definition 4 *Wir sprechen von einem* ***passiven Angreifer****, wenn gilt:*

- *Der Angreifer kann die Kommunikation abhören.*

Definition 5 *Wir sprechen von einem* ***aktiven Angreifer****, wenn zusätzlich gilt:*

- *Der Angreifer kann Nachrichten abfangen.*
- *Der Angreifer kann eigene Nachrichten einschleusen.*

Da die diversen Protokolle allesamt dieselben mathematischen Grundlagen voraussetzen, wollen wir diese im nächsten Abschnitt zusammentragen. Spezielle Grundlagen, wie etwa Signaturverfahren, werden dann jedoch erst besprochen, wenn diese konkret benötigt werden.

1.4 Mathematische Grundlagen

Definition 6 *Sei $(G, \cdot)$ eine Gruppe mit neutralem Element e. Dann heißt*

1. *die Kardinalität $|G|$ die **Ordnung von** G;*
2. *$\mathrm{ord}(g) := \min\{k \in \mathbb{N} : \ g^k = e\}$ die **Ordnung von** $g \in G$.*

Definition 7 *Sei $(G, \cdot)$ eine Gruppe und $g \in G$. Mit $\langle g \rangle$ bezeichnen wir die Untergruppe*

$$\{g^k \mid k \in \mathbb{Z}\} =: \langle g \rangle$$

*für $g^0 := e$ und $g^{z-1} := g^z \cdot g^{-1}$. Von einem Element erzeugte Gruppen $G = \langle g \rangle$ nennen wir **zyklisch**.*

Definition 8 *Die Umkehrabbildung $\log_g : \langle g \rangle \to \mathbb{Z}/n\mathbb{Z}$ des Gruppenisomorphismus*

$$\mathbb{Z}/n\mathbb{Z} \ni [z] \mapsto g^z \in \langle g \rangle$$

*für $\mathrm{ord}(g) = n$ heißt **diskreter Logarithmus** (zur Basis g).*

Beispiel 1 *Ist $(G, \cdot) = \mathbb{F}_{11}^{\times} = \langle g \rangle$ die prime Restklassengruppe modulo 11, so gilt also z. B.*

$$a = \log_g(q) \Leftrightarrow q \equiv g^a \mod 11.$$

Definition 9 *Sei $f : X \to f(X) \subset Y$ eine beliebige Funktion zwischen Mengen X, Y. Wir nennen f eine **Einwegfunktion**, falls f effizient berechenbar, f^{-1} hingegen (auch im Durchschnittsfall) nicht effizient berechenbar ist.*

Annahme 1 *Im folgenden wollen wir annehmen, dass der diskrete Logarithmus eine Einwegfunktion ist (diskretes Logarithmus Problem, kurz DLP)*

Definition 10 *Sei $(\langle g \rangle, \cdot)$ eine zyklische Gruppe der Ordnung n und $a, b, c \in \mathbb{Z}/n\mathbb{Z}$ zufällig gewählt.*
*Die **Computional Diffie-Hellman Hypothese (CDH)** besagt, dass es nicht effizient möglich ist, aus (g, g^a, g^b) den Wert g^{ab} zu berechnen.*
*Die **Decisional Diffie-Hellman Hypothese (DDH)** besagt, dass (g^a, g^b, g^{ab}) und (g^a, g^b, g^c) ununterscheidbar sind.*

Bemerkung 1 *Die Lösbarkeit von CDH und DDH hängt zum einen von der Wahl von G und zum anderen von der Lösbarkeit von DLP ab.*

Bemerkung 2 *Im folgenden werden alle Aussagen und Beweise zur Sicherheit einzelner Protokolle modulo der Annahme geführt, dass DLP, CDH und DDH nicht effizient lösbar sind.*

2 Diffie-Hellman Key Agreement Scheme

Das **Diffie-Hellman KAS** ist ein Public-Key KAS und wurde 1976 von Martin Hellman und Whitfield Diffie zur Vereinbarung eines Schlüssels K zwischen zwei Parteien U und V über einen unsicheren Kanal entwickelt.

2.1 Protokoll

Protokoll 1 *Öffentlich:* $(\langle\alpha\rangle, \cdot)$ *mit* $\mathrm{ord}(\alpha) = n$.

1. *U wählt eine Zufallszahl* $a_U \leq n-1$, *berechnet*

$$b_U = \alpha^{a_U}$$

   und sendet b_U *an* V.

2. *V wählt eine Zufallszahl* $a_V \leq n-1$, *berechnet*

$$b_V = \alpha^{a_V}$$

   und sendet b_V *an* U.

3. *U berechnet* $K = (b_V)^{a_U}$, *V berechnet* $K = (b_U)^{a_V}$.

2.2 Beispiel

Wir wollen uns ein konkretes Beispiel ansehen.

Beispiel 2 *Öffentlich:* $(\langle[\mathbf{2}]_{\mathbf{13}}\rangle, \cdot)$ *mit* $\mathrm{ord}([\mathbf{2}]_{\mathbf{13}}) = \mathbf{12}$.

1. *U wählt eine Zufallszahl* $\mathbf{5} \leq \mathbf{12} - 1$, *berechnet*

$$b_U = \mathbf{6}$$

   und sendet b_U *an* V.

2. *V wählt eine Zufallszahl* $\mathbf{7} \leq \mathbf{12} - 1$, *berechnet*

$$b_V = \mathbf{11}$$

   und sendet b_V *an* U.

3. *U berechnet* $K = ([\mathbf{11}]_{\mathbf{13}})^{\mathbf{5}} = \mathbf{7}$, *$V$ berechnet* $K = ([\mathbf{6}]_{\mathbf{13}})^{\mathbf{7}} = \mathbf{7}$.

2.3 Sicherheit

Modulo CDH und DDH kann ein Angreifer W offensichtlich keine Informationen gewinnen. Das Problem besteht jedoch in einer **Intruder-In-The-Middle-Attack**, bei welcher ein aktiver Angreifer W seine Eigenen Parameter a'_U, a'_V wiefolgt in den Ablauf einbringt:

- U sendet α^{a_U} an V.
- W fängt diesen Wert ab und sendet stattdessen $\alpha^{a'_U}$ an V.
- V sendet α^{a_V} an U.
- W fängt diesen Wert ab und sendet stattdessen $\alpha^{a'_V}$ an U.

Bei einem solchen Angriff ist W also ein tatsächlich ein aktiver Angreifer gemäß der Definition aus Abschnitt 1. Das Ergebnis dieser Attacke ist, dass U einen Schlüssel $K_U = \alpha^{a_U a'_U}$ und V einen Schlüssel $K_V = \alpha^{a_V a'_V}$ besitzen, welche dem Angreifer W beide bekannt sind. D. h. U und V gehen der Annahme nach, dass sie erfolgreich einen Schlüssel ausgetauscht hätten, wobei sie dies jedoch in Wirklichkeit nur mit W getan haben.

Bemerkung 3 *Das Diffie-Hellman-KAS ist nicht sicher gegen eine Intruder-In-The-Middle-Attack.*

3 Station-to-Station Key Agreement Scheme

Um diesem Angriffspunkt des Schemas von Diffie und Hellman zu begegnen, erweitert man es um ein Signaturschema, wodurch die Nachrichten der beiden Teilnehmer authentifiziert werden können.

3.1 Signaturen mit El Gamal

Als Signaturschema kann man zum Beispiel das El-Gamal-Kryptosystem verwenden, welches wie der Diffie-Hellman-Schlüsselaustausch auf das Problem des diskreten Logarithmus aufbaut.

3.1.1 Signaturschemata

Zuerst definieren wir allgemein, wie ein Signaturschema aussieht:

Definition 11 *Ein **Signaturschema** besteht aus:*

- *Nachrichtenmenge $\mathcal{P}$*
- *Signaturenmenge $\mathcal{A}$*
- *Schlüsselmenge $\mathcal{K}$*
- *für $K \in \mathcal{K}$ Signaturfunktionen $sig_K : \mathcal{P} \to \mathcal{A}$*
- *für $K \in \mathcal{K}$ Verifikationsfunktionen $ver_K : \mathcal{P} \times \mathcal{A} \to \{\mathtt{w}, \mathtt{f}\}$*

Dabei soll gelten:

$$ver_K(x, y) = \begin{cases} \mathtt{w} & \text{falls } y = sig_K(x) \\ \mathtt{f} & \text{falls } y \neq sig_K(x) \end{cases}$$

Hierbei interpretieren wir `w` als wahr, also erfolgreiche Verifikation, und `f` als falsch, also fehlgeschlagene Verifikation.

3.1.2 El Gamal

Das Signaturschema nach El Gamal sieht nun folgendermaßen aus:

Definition 12 *Sei p Primzahl und erzeugendes Element $\alpha \in \mathbb{Z}_p^\times$. Dann ist*

$$\mathcal{K} := \{(p, \alpha, a, \beta) \mid \beta \equiv_p \alpha^a\}$$

*die **Schlüsselmenge**, wobei p, α und β öffentlich sind; a ist geheim.*

Sei $x \in \mathcal{P} = \mathbb{Z}_p^\times$ die Nachricht[1], dann definieren wir:

Definition 13 *Seien $k \in \mathbb{Z}_{p-1}^\times$ Zufallszahl mit $\mathrm{ggT}(k, p-1) = 1$, $\mathcal{A} := \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$ dann ist*

$$sig_K(x, k) = (\gamma, \delta)$$

***Signaturfunktion** mit $\gamma = \alpha^k$ und $\delta = (x - a\gamma)k^{-1} \bmod (p-1)$*

[1] An dieser Stelle wird in der Praxis in der Regel eine kryptographische Hashfunktion auf die Nachricht ausgeführt und der resultierende Hash signiert.

Definition 14 *Als* ***Verifikationsfunktion*** *definieren wir:*

$$ver_K(x, (\gamma, \delta)) = \mathtt{w} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv_p \alpha^x$$

Um zu zeigen, dass diese Definition die gewünschten Eigenschaften erfüllt, betrachten wir die Verfikationsgleichung

$$\alpha^x \equiv_p \beta^\gamma \cdot \gamma^\delta$$

und substituieren $\gamma \equiv_p \alpha^k$ und $\beta \equiv_p \alpha^a$. Es ergibt sich:

$$\alpha^x \equiv_p \alpha^{a\gamma + k\delta}$$

Da α erzeugendes Element der Gruppe ist, muss gelten:

$$x \equiv_{p-1} a\gamma + k\delta$$

Nach δ umgestellt ergibt sich:

$$\delta \equiv_{p-1} (x - a\gamma) \cdot k^{-1}$$

3.1.3 Sicherheit

Das El-Gamal-Kryptosystem wird als sicher angesehen. Ein Angreifer hat grundsätzlich zwei Möglichkeiten, zu versuchen, eine Signatur zu fälschen:

1. Der Angreifer wählt γ und will passendes $\delta = \log_\gamma(\alpha^x \beta^{-\gamma})$ berechnen: Diskreter Logarithmus.
2. Der Angreifer wählt δ und will passendes γ berechnen: Auflösung von $\beta^\gamma \gamma^\delta \equiv_p \alpha^x$ nach γ, wozu auch kein Verfahren bekannt ist.

3.2 STS KAS

Nun sind alle Voraussetzungen erfüllt, um das Staton-to-station KAS zu definieren. Dazu bekommt jeder Teilnehmer U von der TA:

- Signaturfunktion sig_U
- Verifikationsfunktion ver_U
- Identifikation $ID(U)$
- Zertifikat $\mathbf{Cert}(U) = (ID(U), ver_U, sig_{TA}(ID(U), ver_U))$

Das Protokoll entspricht nun dem Diffie-Hellman-Schlüsseltausch mit der Erweiterung, dass zu den getauschten Zahlen zusätzlich Signaturen verschickt werden, die eine Manipulation verhindern sollen.

Protokoll 2 *Öffentlich:* $(\langle\alpha\rangle, \cdot)$ *mit* $\mathrm{ord}(\alpha) = n$.

1. *U wählt eine Zufallszahl $a_U \leq n - 1$ und berechnet:*

$$b_U = \alpha^{a_U}$$

Er sendet $\mathbf{Cert}(U)$ *und* b_U *an* V.

2. *V wählt eine Zufallszahl $a_V \leq n-1$ und berechnet:*

$$b_V = \alpha^{a_V}, \quad K = (b_U)^{a_V}, \quad y_V = sig_V(ID(U)||b_V||b_U)$$

Sie sendet **Cert**(V)*, b_V und y_V an U.*

3. *U verifiziert y_V mit ver_V. Wenn dies erfolgreich ist, berechnet er:*

$$K = (b_V)^{a_U}, \quad y_U = sig_V(ID(V)||b_U||b_V)$$

Er sendet y_U an V.

4. *V verifiziert y_U mit ver_U.*

3.3 Definitionen zur Sicherheit

Definition 15 ***implizite Schlüsselauthentifikation*** *(implicit key authentication):*

- *U weiß, dass niemand außer V den Schlüssel berechnen kann.*

Definition 16 ***implizite Schlüsselbestätigung*** *(implicit key confirmation):*

- *U weiß, dass niemand außer V den Schlüssel berechnen kann.*
- *U weiß, dass V den Schlüssel berechnen kann.*

Definition 17 ***explizite Schlüsselbestätigung*** *(explicit key confirmation):*

- *U weiß, dass nur V den Schlüssel berechnen kann.*
- *U weiß, dass V den Schlüssel berechnet hat.*

3.3.1 Der Angriff aus der Mitte

Wie beim Diffie-Hellman-Schlüsseltausch kann ein passiver Angreifer nicht die Schlüssel berechnen, da er dazu das CDH lösen müsste.
Es ist leicht zu sehen, dass auch ein aktiver Angreifer nicht durch eingeschleuste Nachrichten falsche Schlüssel einbringen kann, denn er müsste bei Nachrichten an U die Signatur von V fälschen und umgekehrt.

- Perspektive: U
  Wenn U von V b_V und $y_V = sig_V(ID(U)||b_V||b_U)$ erhalten hat, kann er sicher sein, dass

  b_V wirklich von V ist, da Signatur y_V stimmt

  kein Angreifer $K = (b_U)^{a_V}$ berechnen kann, da er a_V nicht kennt

  V selbst $K = (b_U)^{a_V}$ berechnen kann, da V das richtige b_U und a_V kennt.

- Perspektive: V
  Wenn V von U b_U und $y_U = sig_U(ID(V)||b_U||b_V)$ erhalten hat, kann er sicher sein, dass

  b_U wirklich von U ist, da Signatur y_U stimmt

  kein Angreifer $K = (b_V)^{a_U}$ berechnen kann, da er a_U nicht kennt

  V selbst $K = (b_V)^{a_U}$ berechnen kann, da V das richtige b_V und a_V kennt.

Es besteht allerdings eine gewisse Asymmetrie, denn wenn V akzeptiert, weiß er, dass U schon akzeptiert hat. Umgekehrt gilt dies nicht.
Aus diesen Erkenntnissen lässt sich der Satz formulieren:

Satz 1 *Station-to-station KAS ist ein authenticated KAS, welches*

- *implizite Schlüsselbestätigung für beide Teilnehmer bietet,*

wenn das DDH-Problem praktisch nicht lösbar ist.

3.3.2 Angriffe mit bekannten Sitzungsschlüsseln

Definition 18 *Wir sprechen von einem Known-Session-Key-Angriff, wenn*

- *der Angreifer mehrere Sitzungen von Nutzern im Netzwerk beobachten*
- *und sogar selbst an Sitzungen mit U und V teilnehmen kann.*

Wenn ein Angreifer O (Oscar) (O) einen KSK-Angriff auf STS versucht, ist leicht zu sehen, dass bei einer Sitzung mit einem beliebigen Teilnehmer W gilt:

- Bevor Oscar den Schlüssel berechnen kann, gewinnt er keine Informationen, die nicht auch ein passiver Angreifer hätte.
- Sobald Oscar den Schlüssel berechnen kann, erhält er eine Lösung des **CDH**-Problems: $(b_O, b_W, K = \mathbf{CDH}(b_O, b_W))$

Nun verfolgen wir folgende Idee, um die Sicherheit von STS gegenüber KSK-Angriffen zu zeigen:

- Oscar nimmt an Sitzung $\mathcal{S}_i$ mit einem beliebigen Teilnehmer W teil und erhält das Tripel $t_i = (b_{\mathcal{S}_i,O}, b_{\mathcal{S}_i,W}, K_{\mathcal{S}_i})$, wobei $K_{\mathcal{S}_i} = \mathbf{CDH}(b_{\mathcal{S}_i,O}, b_{\mathcal{S}_i,W})$ darstellt.
- Oscar sammelt aus Sitzungen $\mathcal{S}_0 \dots \mathcal{S}_n$ die Tripel $t_0 \dots t_n$ und erzeugt Transskript $T = \{t_i \mid i \leq n\}$.
- Oscar versucht, aus T Informationen über den Schlüssel $K_{\mathcal{S}}$ einer Sitzung $\mathcal{S}$ zu gewinnen, an der er nicht teilnimmt.
- Wir nehmen an, Oscar hat dafür einen (effizienten) Algorithmus A und führen diese Annahme zum Widerspruch.

Oscar könnte ein Tripel $t_{sim} = (b_{sim,O}, b_{sim,W}, K_{sim})$ erzeugen, ohne an einer Sitzung teilzunehmen, indem er:

- $b_{sim,O}$ wählt wie bisher.
- zufällig a_W wählt und $b_{sim,W} = \alpha^{a_W}$ berechnet.
- $K_{sim} = (b_{sim,W})^{a_O}$ berechnet.

Da der einzige Unterschied darin besteht, dass die zufällige Wahl von $b_{sim,W}$ durch W durch die zufällige Wahl von Oscar ersetzt wurde, ist leicht zu sehen, dass t_{sim} nicht von einem Tripel t unterscheidbar ist, das aus einer Sitzung gewonnen wurde.

Aufbauend darauf kann Oscar ganze simulierte Transskripte erzeugen, die wiederum nicht von echten Sitzungstransskripen unterschieden werden können, insbesondere nicht von Oscars hypothetischem Algorithmus A.
Da es keinen Unterschied macht, ob Oscar A auf ein echtes oder ein simuliertes Transskript ansetzt, kann der ganze Angriff vollkommen passiv durchgeführt werden.
Dies ist ein Widerspruch zu der Annahme, dass es nicht möglich ist, als passiver Angreifer Informationen über $K_{\mathcal{S}}$ zu gewinnen, wenn das DDH-Problem nicht lösbar ist. Somit kann der Algorithmus A nicht existieren.
Daraus ergibt sich folgender Satz:

Satz 2 *Station-to-station KAS ist ein authenticated KAS, welches*

- *implizite Schlüsselbestätigung für beide Teilnehmer bietet und*
- *sicher gegen known-session-key-Angriffe ist,*

wenn das DDH-Problem praktisch nicht lösbar ist.

4 MTI Key Agreement Scheme

Ein weiteres Protokoll der Public-Key-Familie ist das **MTI KAS** entwickelt von T. Matsumoto, Y. Takashima und H. Imai. Es ist ein Protokoll, welches im Gegensatz zum Station-To-Station KAS keine Authentifizierung und Signierung benötigt, dafür aber auf den Einsatz von Trusted Authorities (TA), d. h. gewissen Zertifizierungsstellen, vertraut. Wir betrachten nun ein spezielles Schema der MTI-Familie: MTI/A0.

4.1 Protokoll MTI/A0

Protokoll 3 *Öffentlich:* $(\langle\alpha\rangle, \cdot)$ *mit* $\mathrm{ord}(\alpha) = n$.

1. *U wählt Zufallszahlen* $a_U, r_U \leq n-1$, *berechnet*

$$\begin{aligned} b_U &= \alpha^{a_U} \hookrightarrow_{\mathrm{sig}_{\mathrm{TA}}} \mathbf{Cert}(U) \\ s_U &= \alpha^{r_U} \text{ (je Session neu)} \end{aligned}$$

   und sendet $\mathbf{Cert}(U)$ *und* s_U *an* V.

2. *V wählt eine Zufallszahl* $a_V, r_V \leq n-1$, *berechnet*

$$\begin{aligned} b_V &= \alpha^{a_V} \hookrightarrow_{\mathrm{sig}_{\mathrm{TA}}} \mathbf{Cert}(V) \\ s_V &= \alpha^{r_V} \text{ (je Session neu)} \end{aligned}$$

   und sendet $\mathbf{Cert}(V)$ *und* s_V *an* U.

3. *U berechnet* $K = (s_V)^{a_U}(b_V)^{r_U} = \alpha^{r_U a_V + r_V a_U}$,
   V berechnet $K = (s_U)^{a_V}(b_U)^{r_V} = \alpha^{r_U a_V + r_V a_U}$.

Hierbei bezeichnen $\mathbf{Cert}(U)$ und $\mathbf{Cert}(V)$ die von der TA signierten Zertifikate für U und V, welche die Werte b_U bzw. b_V enthalten. Eine Protokollinstanz nennen wir auch **Session**. Je Session werden die s_U, s_V neu durch U und V berechnet. Die Zertifikate jedoch können durchaus über einen längeren Zeitraum bestehen.

4.2 Sicherheit

4.2.1 Intruder-In-The-Middle-Attack

Das Protokoll hält offensichtlich einer Intruder-In-The-Middle-Attack stand. Angenommen W verfahre wiefolgt:

- U sendet $(\mathbf{Cert}(U), \alpha^{r_U})$ an V.
- W fängt diesen Wert ab und sendet stattdessen $(\mathbf{Cert}(U), \alpha^{r'_U})$ an V.
- V sendet $(\mathbf{Cert}(V), \alpha^{r_V})$ an U.
- W fängt diesen Wert ab und sendet stattdessen $(\mathbf{Cert}(V), \alpha^{r'_V})$ an U.

Da W keine Möglichkeit hat, die Zertifikate zu ändern, würden U und V folglich folgende Schlüssel generieren:

$$\begin{aligned} K_U &= \alpha^{r_U a_V + r'_V a_U}, \\ K_V &= \alpha^{r'_U a_V + r_V a_U}. \end{aligned}$$

W kann sowohl keinen dieser Schlüssel berechnen, noch kann er Informationen daraus ziehen (modulo DDH).

4.2.2 Parallel-Session-Key-Attack

Angenommen W partizipiert nun in zwei verschiedenen Sessions $\mathcal{S}$ und $\mathcal{S}'$ des MTI/A0-Protokolls. In beiden Session agiere W als Akzeptor: zum einen mit V in Session $\mathcal{S}'$ und zum anderen mit U in Session $\mathcal{S}$. D. h. W gibt in Session $\mathcal{S}$ vor, U und andererseits in $\mathcal{S}'$ V zu sein. Die Interaktion liefe dann zeitlich in folgenden vier Schritten ab:

1. V sendet $(\mathbf{Cert}(V), s_V)$ an W (Session $\mathcal{S}'$).
2. U sendet $(\mathbf{Cert}(U), s_U)$ an W (Session $\mathcal{S}$).
3. W sendet $(\mathbf{Cert}(V), s_V)$ an U (Session $\mathcal{S}$).
4. W sendet $(\mathbf{Cert}(U), s_U)$ an V (Session $\mathcal{S}'$).

Wie man sich nun leicht überlegt, sind die beiden Schlüssel in den Sessions $\mathcal{S}$ und $\mathcal{S}'$ gleich:

$$K_{\mathcal{S}'} = \alpha^{r_U a_V + r_V a_U} = \alpha^{r_V a_U + r_U a_V} = K_{\mathcal{S}}.$$

Als Konsequenz erhält W vollständige Information über einen vereinbarten Schlüssel K. Da dieser Angriff nur aufgrund der Symmetrie der Schlüsselgenerierung möglich ist, bestünde eine zweckmäßige Verbesserung des Protokolls darin, die Schlüssel mithilfe einer asymmetrischen Hash-Funktion h zu encodieren:

$$K_{\mathcal{S}'} = h\left(\alpha^{r_U a_V} || \alpha^{r_V a_U}\right) \neq h\left(\alpha^{r_U a_V} || \alpha^{r_V a_U}\right) = K_{\mathcal{S}}$$

aber

$$K_U = h\left(\alpha^{r_U a_V} || \alpha^{r_V a_U}\right) = K_V.$$

Hierbei bezeichne der Operator $||$ die Konkatenation von Strings.

4.2.3 Burmester Triangle Attack

Eine weitere Möglichkeit, gegen das MTI/A0 Protokoll vorzugehen, ist die sogenannte **Burmester Triangle Attack**. Innerhalb dieses Angriffs, kann der Angreifer W den Schlüssel $K_{\mathcal{S}}$ ermitteln, ohne aktiv an dieser Session zu partizipieren. Dieser Angriff ist ein Known-Session-Key-Angriff:

I. Session $\mathcal{S}$: W observiert einen Schlüsselaustausch zwischen U und V. Diesbezüglich erhält W die Daten: $(\mathbf{Cert}(U), s_U)$ und $(\mathbf{Cert}(V), s_V)$.

II. Session $\mathcal{S}_1$:

1. W sendet $(\mathbf{Cert}(W), s_U)$ an V.
2. V sendet $(\mathbf{Cert}(V), s'_V)$ an W.

III. Session $\mathcal{S}_2$:

1. W sendet $(\mathbf{Cert}(W), s_V)$ an U.
2. U sendet $(\mathbf{Cert}(U), s'_U)$ an W.

Als Resultat kennt nun W zwei Schlüssel $K_{\mathcal{S}_1}$ und $K_{\mathcal{S}_2}$:

$$\begin{aligned} K_{\mathcal{S}_1} &= \alpha^{r_U a_V + r'_V a_W}, \\ K_{\mathcal{S}_2} &= \alpha^{r'_U a_W + r_V a_U}. \end{aligned}$$

Damit stellt die Burmester Triangle Attack einen erfolgreichen Angriff auf das MTI/A0 Protokoll dar, denn es gilt der

Satz 3 *In der Burmester Triangle Attack auf das* MTI/A0 *KAS mit U, V als Parteien und W als Angreifer, gilt*

$$K_{\mathcal{S}} = K_{\mathcal{S}_1} \cdot K_{\mathcal{S}_2} \cdot \left((s'_V s'_U)^{a_W}\right)^{-1}$$

für die Gruppenoperation „$\cdot$“. Es gelten die Bezeichnungen aus dem Protokoll MTI/A0*.*

Beweis: laut Voraussetzung gilt

$$\begin{aligned} K_{\mathcal{S}_1} &= \alpha^{r_U a_V + r'_V a_W}, \\ K_{\mathcal{S}_2} &= \alpha^{r'_U a_W + r_V a_U}. \end{aligned}$$

Damit erhalten wir

$$\begin{aligned} K_{\mathcal{S}_1} \cdot K_{\mathcal{S}_2} &= \alpha^{r_U a_V + r'_V a_W} \cdot \alpha^{r'_U a_W + r_V a_U} \\ &= \alpha^{r_U a_V + r'_V a_W + r'_U a_W + r_V a_U} \\ &= \underbrace{\alpha^{(r'_U + r'_V) a_W}}_{\left(\alpha^{(r'_U + r'_V)}\right)^{a_W}} \cdot \underbrace{\alpha^{r_U a_V + r_V a_U}}_{K_{\mathcal{S}}} \\ &= (s'_V s'_U)^{a_W} \cdot K_{\mathcal{S}} \end{aligned}$$

und dies zeigt die Behauptung. □

Bemerkung 4 *Das MTI/A0 Protokoll ist sicher gegen eine Intruder-In-The-Middle-Attack aber nicht sicher gegen Parallel-Session-Key-Attacks und Burmester Triangle Attacks. Diese Unsicherheit kann aber durch Einführung einer Hash-Funktion beseitigt werden.*

5 Self-certifying keys

Eine weitere Möglichkeit, Schlüsselvereinbarung kryptographisch umzusetzen, bieten self-certifying keys. Hierzu wird auf das RSA-Kryptosystem zurückgegriffen.

5.1 Mathematische Grundlagen

Sei die eulersche φ-Funktion definiert durch

$$\varphi(n) := |\{1 \leq a \leq n \mid \text{ggT}(a, n) = 1\}|$$

$p, q \in \mathbb{N}$ Primzahlen und $n = p \cdot q$, dann gilt:

- $\varphi(n) = (p-1) \cdot (q-1)$
- $x^{\varphi(n)} \equiv_n 1$ (Satz von Euler)

Für $e, d < \varphi(n)$ mit $e \cdot d \equiv_{\varphi(n)} 1$ gilt:

- $(x^e)^d \equiv_n x$ (f. a. x, die Teilerfremd zu n sind), da $x^{ed} = x^{1+l\cdot\varphi(n)} = x \cdot (x^{\varphi(n)})^l$.

Annahmen:

- $\varphi(n)$ lässt sich ohne Kenntnis von p und q nicht effizient berechnen.
- $\sqrt[e]{x^e}$ lässt sich nicht effizient berechnen.

5.2 Vorbereitungen der TA

Die Trusted Authority muss sich zur Vorbereitung einen Modul n und die dazu passenden Exponenten erzeugen, die für die selbstzertifizierenden Schlüssel benötigt werden.

- Wähle zufällig große Primzahlen $p, q \in \mathbb{N}$
- Berechne $n = p \cdot q$
- Berechne $\varphi(n) = (p-1) \cdot (q-1)$
- Wähle e, so dass $1 < e < \varphi(n)$ und $\text{ggT}(e, \varphi(n)) = 1$.
- Berechne $d = e^{-1} \bmod \varphi(n)$

n, e sind öffentlich, d ist geheim.

5.2.1 Hinweis:

Alle übrigen Zahlen werden nicht weiter benötigt. In der Praxis sollten diese sicher gelöscht werden.

5.3 Schlüsselgenerierung

Jeder Nutzer muss nun mit der TA seinen Schlüssel generieren, im späteren Verlauf ist dann kein Einschreiten der TA mehr notwendig. Die Schlüsselgenerierung ist der einzige Teil des Protokolls, der einen (gegen Abhören und Manipulation) sicheren Kanal erfordert.

Protokoll 4 *α, n sind öffentlich, d ist nur der TA bekannt.*

1. *U wählt geheimen Exponenten a_U und berechnet*

$$b_U = \alpha^{a_U} \bmod n$$

   U schickt a_U und b_U an die TA.

2. *Die TA berechnet*

$$p_U = (b_U - ID(U))^d \bmod n$$

   Die TA schickt p_U an U.

Wir werden später sehen, dass es für die Sicherheit wichtig ist, dass die TA von U auch a_U erhält und überprüft, auch wenn a_U für die Berechnungen eigentlich nicht notwendig ist.

5.4 Protokoll

Das Protokoll läuft nun folgendermaßen ab:

Protokoll 5 *n, e, α sind öffentlich.*

1. *U wählt zufällig r_U und berechnet*

$$s_U = \alpha^{r_U} \bmod n$$

   U sendet $ID(U)$, p_U und s_U an V.

2. *V wählt zufällig r_U und berechnet*

$$s_V = \alpha^{r_V} \bmod n$$

   V sendet $ID(V)$, p_V und s_V an U.

3. *U berechnet*

$$K = {s_V}^{a_U} ({p_V}^e + ID(V))^{r_U} \bmod n$$

   V berechnet

$$K = {s_U}^{a_V} ({p_U}^e + ID(U))^{r_V} \bmod n$$

5.5 Angreifer gibt falsche Identität vor

b_U, p_U und $ID(U)$ sind nicht von der TA signiert.

- W kann falsches p'_U wählen und ein passendes b'_U berechnen.
- *Aber:* W kann nicht das passende $a_U = \log_\alpha b_U$ berechnen (diskreter Logartihmus).

Beobachtung: Das selbe Problem hat W, wenn er als intruder-in-the-middle agiert.

Beobachtung: Dieses KAS bietet implizite Schlüsselauthentifikation.

5.6 Unachtsame TA

Wenn die TA nicht sicherstellt, dass ein Nutzer U den Wert a_U kennt, ist das Schema angreifbar:

- W wählt falsches a'_U und berechnet $b'_U = \alpha^{a'_U}$
- Er berechnet $b'_W = b'_U - ID(U) + ID(W)$
- Die TA brechnet für ihn $p'_W = (b'_W - ID(W))^d \bmod n$

Da $b'_W - ID(W) \equiv_n b'_U - ID(U)$, gilt $p'_W = p'_U$

- W hat jetzt einen Schlüssel, der von U zu sein scheint
- und kennt den passenden Exponenten a'_U.

Damit kann W eine Sitzung abhalten, in der sein Gegenüber ihn für U hält.

Literatur

[1] Douglas Robert Stinson, „Cryptography. Theory and Practice.“, 3rd ed. 2006, Chapman & Hall/CRC.

[2] Wikipedia (DE, EN), Stand: 4. März 2009.