MQV Protokoll

MQV Protokolle:

- Schlüsselberechnung symmetrisch.
- w_A, W_A, w_B, W_B private/öffentliche Schlüssel.
- Geheime Zufallszahlen r_A, r_B. $R_A = r_A P$ und $R_B = r_B P$ werden von A und B ausgetauscht.
- A berechnet $s_A = (r_A + \mu(R_A) w_A) \bmod \ell$, $S_B = (R_B + \mu(R_B) W_B)$ und $K_{AB} = h s_A S_B$. B analog mit A und B vertauscht.

Two-Pass Variante: AK, keine Key Confirmation.

One-Pass Variante von A nach B: Verwendet w_B, W_B anstelle von r_B, R_B. AK, keine Key Confirmation. Keine known-key oder forward secrecy, da B passiv ist.

Unknown Key-Share möglich (durch Zusatz leicht verhinderbar).

Standards

Ziel: Interoperabilität, sichere Implementierungen.

Zu definieren:

- Format der Daten, Körperelemente, Punkte, Signaturen, (Public-Key) Chiffretexte, ...
- Konversionsfunktionen dieser Objekte nach Bitstring, Integers, ...
- Genauer Spezifikation der Algorithmen, Punktkompression ja/nein, Hybridverschlüsselung, ...

Ein paar Standards/Organisationen:

- International: IEEE, IETF, ISO, ITU.
- National: ANSI, CEN, DIN, NIST.
- Company: PKCS, SECG.

Standards

IEEE P1363/P1363a: Enthält alle Public-Key Verfahren.

- EC-DH, EC-DSA, EC-MQV, EC-IES.
- Anhang enthält viele grundlegende, zahlentheoretische Algorithmen.

ANSI X9.62: EC-DSA.
ANSI X9.63: EC-DH, EC-MQV, EC-IES.

FIPS186.2: NIST Standard für DSA und EC-DSA.

SECG: Indutriestandard geführt von Certicom.

- EC-DH, EC-DSA, EC-MQV, EC-IES.

IETF: IPsec, S/MIME, ...

ISO/IEC 14888-3: Identity-based cryptography.

PKCS #13: ECC.

Standards

Alle (außer IEEE) geben Nachrichtenformate und empfohlene Kurven an.

- Nicht jeder kann/will geeignete Körper und Kurven selber finden (Punktezählalgorithmus implementieren, ...)
- Unterstützt Verbreitung und Interoperabilität.

Verschiedene Standards benutzen teilweise die gleichen Kurven.

- Unterstützt Verbreitung und Interoperabilität.

Die ANSI und NIST Kurven sind „am besten“.

Arbeitsweise von Standards:

- „Freiwillige“ Mitarbeit von Firmenangestellten. Wenig Academia.
- Dadurch Interessensvertretung der Firmen ...

Identifikation

Eine logische Einheit will sich gegenüber einem System identifizieren, um Zugang, Resourcen, Information, Kontrolle etc. zu erhalten.

Beispiel: Geld abheben vom Konto (in der Bank oder vom Geldautomat), einloggen, ...

Der Ansatz ist, Identität mit einer geheimen Information bzw. Wissen gleichzusetzen: PIN, Paßwort, geheimer Schlüssel, Kenntnis des geheimen Schlüssels zu einem öffentlichem Schlüssel.

Dies führt auf Identifikationsprotokolle, in denen der Beweiser einen Verifizierer von seiner Identität überzeugen möchte.

Paßwörter

Hier benutzt man eine Einwegfunktion f. Der Beweiser offenbart dem Verifizierer sein Paßwort w, dieser berechnet $f(w)$ und überprüft, ob der Wert in der Benutzer-Datei für den Beweiser vorkommt.

Gefahren:

- Paßwort wird bekannt durch Verschulden des Verifizierers oder sonstige technische Probleme (Paßworteingabefenster ist gar nicht das „echte“).
- Dictionary Attack (Paßwörter intelligent raten).
- Paßwort kann dann durch erfolgreichen Angreifer verwendet werden.

Einmal-Paßwörter

Verbesserung gegenüber Paßwörtern: Jedes Paßwort wird nur einmal verwendet.

Beispiel: TAN beim online Banking.

Einmal-Paßwörter mittels Einwegfunktion H.

Initialisierung:

- Beweiser wählt geheimes w_0 und setzt $w_t = H^t(w_0)$.
- w_t wird authentisch (!) zum Verifizierer transportiert.
- Beweiser und Verifizierer halten einen Zähler i über die erfolgten Identifikationen.
- Der Verifizierer merkt sich w_{t-i} nach der i-ten Verifikation.

Einmal-Paßwörter

Identifikation:

- In der $(i+1)$-ten Identifikation berechnet der Beweiser $w_{t-i-1} = H^{t-i-1}(w_0)$ und schickt w_{t-i-1} an den Verifizierer. Dieser überprüft $w_{t-i} = H(w_{t-i-1})$.

Eigenschaften:

- Wird ein w_{t-i} bekannt, so bleiben die nachfolgenden sicher.
- Begrenzte Anzahl von Passwörtern.
- Nicht simultan für verschiedene Identifikationen benutzbar, ohne Replayangriff zu ermöglichen.

Challenge-Response Identifikation (nachfolgend) ist da flexibler.

Challenge-Response Identifikation

Einfaches Beispiel 1:

- Verifizierer schickt Beweiser eine geeignete (zufällige) Nachricht.
- Beweiser schickt eine Signatur der Nachricht zurück.
- Verifizierer überprüft die Signatur.

Einfaches Beispiel 2:

- Verifizierer schickt Beweiser einen geeigneten (zufälligen) Chiffretext.
- Beweiser schickt die Entschlüsselung des Chiffretexts zurück.
- Verifizierer überprüft die Entschlüsselung.

Man-in-the-Middle oder Grandmaster Postal-Chess Problem:

- Jemand kann sich zwischen die kommunizierenden Parteien hängen. Zählt bei C-R-I nicht direkt als Angriff.

Challenge-Response Identifikation

Beweiser sei B und Verifizierer A (B und A werden hier mit den öffentlichen Schlüsseln B und A gleichgesetzt).

Beispiel (Gegenseitige Identifikation):

- B schickt A eine Zufallszahl r_B.
- A schickt B eine Zufallszahl r_A, B und die Signatur $S_A(r_A, r_B, B)$.
- B überprüft die Korrektheit der Daten (also B und die Signatur), A hat sich damit dann gegenüber B identifiziert.
- B schickt A den Wert A und die Signatur $S_B(r_B, r_A, A)$.
- A überprüft die Korrektheit der Daten (also A und die Signatur), B hat sich damit dann gegenüber A identifiziert.

Weitere Protokolle speziell für Identifikation: Zero Knowledge Beweise etc.

Salzen und Strecken

Nochmal zu den Paßwörtern. Im folgenden zwei Techniken, einen Dictionary-Angriff zu erschweren, sollte der Angreifer Zugriff auf die Paßwortdatei mit Einträgen $(\text{ID}_i, H(\text{Passwort}_i))$ haben.

Strecken:

- Statt H einmal anwenden, H sehr häufig anwenden, also Paßwortdatei mit Einträgen der Form $(\text{ID}_i, H^r(\text{Passwort}_i))$ verwenden.
- r sollte so groß sein, daß die Berechnung beispielsweise ca. 1 Sekunde dauert.
- Der Angreifer hat dann einen unumgänglich größeren Rechenaufwand.

Salzen und Strecken

Salzen:

- Zu den Paßwörtern Salz hinzufügen, also Paßwortdatei mit Einträgen $(\text{ID}_i, H^r(\text{Passwort}_i || \text{Salz}_i), \text{Salz}_i)$ verwenden.
- Zum Beispiel $\text{Salz}_i = i$, oder besser Salz_i ein Zufallswert.
- Dictionary-Angriff bezieht sich dann nur auf eine gewählte ID_i, nicht alle zusammen (beachte Geburtstagsparadxon), bzw. wird für alle erschwert.

Zusätzlich:

- Login Prompt hat normalerweise eingebaute Verzögerung, wenn Paßwort falsch eingegeben wird.

Interaktive Beweissysteme

Hier nur informell.

Haben zwei Programme, einen Beweiser P und einen Verifizierer V. Diese tauschen Nachrichten aus, zum Schluß akzeptiert V oder weist zurück.

P und V haben als Eingabe jeweils ein Zufallsband und weitere, gemeinsame Daten. P erhält darüberhinaus weitere (vor V geheimen) Daten, welche Zeuge genannt werden.

Die Idee ist, daß P die Lösung zu einem Problem erhält und V davon überzeugen muß, daß es die Lösung kennt, ohne etwas über die Lösung zu verraten.

Allgemein betrachten wir (P,V) als Programm mit zwei kommunizierenden, beliebigen Teilprogrammen P und V.

Interaktive Beweissysteme

Beispiel: Für ein y kennt P ein x mit $y = g^x$. V kennt nur y. P will beweisen, daß es x kennt.

- P könnte x an V schicken, aber dann wäre es V bekannt.
- Abgesehen davon gibt es Probleme und Lösungen x, wo es schwierig ist, zu prüfen, ob x eine Lösung ist. Betrachten wir hier nicht.

Es sollen folgende Eigenschaften gelten:

- Vollständigkeit: Wenn P den Zeugen x als Eingabe erhält (P kennt x), dann akzeptiert V.
- Korrektheit: Für jedes Programm B: Wenn V in der Interaktion mit B akzeptiert, dann hat B den Zeugen mit Wahrscheinlichkeit $\geq 1/3$ als Eingabe erhalten.

Dann heißt (P,V) ein interaktives Beweissystem.

Zero Knowledge Beweise

Daß P nichts weiteres bezüglich x verrät, wird wie folgt formalisiert.

Für jedes Programm V* gibt es ein Programm M*, dessen Ausgaben als Zufallsvariable mit den Ausgaben von (P,V^*) als Zufallsvariable in Abhängigkeit der Zufallsbänder nicht effizient unterschieden werden können (bei gleicher, fixer Eingabe der zusätzlichen Parameter, bis auf den Zeugen).

Die Eingabe des Zeugen soll also für eine (jede) Berechnung keinen Vorteil bringen.

Es genügt, nur solche M* zu betrachten, welche die zwischen P und V* kommunizierten Nachrichten und das Zufallsband von V* ausgeben, da diese als Eingabe von V* aufgefaßt werden können und V* danach deterministisch abläuft. Man nennt M* einen Simulator.

Zero Knowledge Beweise

Sei n RSA Modul, x zufällig und $y = x^2 \bmod n$.
P soll Kenntnis von x beweisen, ohne etwas über x zu verraten.

1. P wählt zufälliges r und schickt $a = r^2$ an V.
2. V wählt zufälliges $e \in \{0,1\}$ und schickt e an P.
3. P berechnet $b = rx^e$ und schickt b an V.
4. V akzeptiert genau dann, wenn $b^2 = ay^e$.

Ein schummelndes Programm B anstelle von P kann wie folgt mit Erfolgswahrscheinlichkeit $1/2$ vorgehen:

1. B wählt zufälliges r und rät e im voraus, also $e' \in \{0,1\}$ zufällig.
2. B berechnet $a = r^2 y^{-e'}$ und schickt a an V.
3. V schickt zufälliges e an B, und B antwortet mit r.

Zero Knowledge Beweise

Angenommen, es gibt ein schummelndes B mit Erfolgswahrscheinlichkeit $> 1/2$.

Dann kann B Wurzeln b, r mit $b^2 = ay^e$ und $r^2 = ay^{e'}$ und $e \neq e'$ finden. Folglich kann B Quadratwurzeln von y ausrechnen, also n faktorisieren.

Durch mehrfache Wiederholung des interaktiven Beweises kann man die Erfolgswahrscheinlichkeit für B beliebig klein machen.

Zero Knowledge Beweise

Simulation des Beweises: Sei V* gegeben. Wir konstruieren M* wie B, nur daß M* noch $e' = V^*(a)$ überprüft und solange rechnet, bis dies gilt.

Die Ausgabe von M* ist dann nicht von der von (P, V^*) zu unterscheiden:

- a ist zufällig, $e = V^*(a)$, und b ist eine korrekte Quadratwurzel.

Damit ist der angegebene interaktive Beweis ein Zero Knowledge Beweis (of Knowledge).

Commitment Schemes

Ziel: P legt sich vor V auf einen Wert fest, ohne daß V den Wert erfährt. Später muß P den Wert bekannt geben.

Eigenschaften:

- Hiding (comput., uncond.): Der Wert ist geheim vor V.
- Binding (comput., uncond.): P kann bei Bekanntgabe nicht schummeln.

Variante basierend auf DLP:

1. Challenge. V schickt g, v an P.
2. Commit. P legt sich auf die Nachricht m wie folgt fest. P wählt zufälliges r und schickt $c = g^r v^m$ an V.
3. Reveal. P schickt m, r an V. V testet $c = g^r v^m$.

Commitment Schemes

Wenn P schummeln könnte, dann hätten wird eine Gleichung $g^r v^m = g^{r'} v^{m'}$ und somit das DLP $v = g^{(r-r')/(m'-m)}$.
P kann $(r - r')/(m' - m)$ berechnen.

Der Wert c ist völlig zufällig, gibt keine Information über m preis.
V hat daher auch dann keinen Vorteil, wenn es s mit $v = g^s$ kennt.

Beispiel: Münzwurf übers Telefon.
Scheme ist unconditionally hiding und computationally binding.

Außerdem:
Mit $C(m, r) = g^r v^m$ gilt $C(r_1, m_1)C(r_2, m_2) = C(r_1 + r_2, m_1 + m_2)$.
$\Rightarrow$ homomorphes Commitment.

Dies findet Anwendung beim verdeckten Rechnen, e-Voting.

Weitere Protokolle

- Multiparty Kryptographie.
- Threshold Kryptographie („Entschlüsseln, wenn die Mehrheit dafür ist“).
- Multi-party computation.

- Oblivious transfer, simulaneous contract signing.
- Auktionen.
- ... ...
- ...

Paarungsbasierte Kryptographie

Seien G_1, G_2, G_T zyklische Gruppen von Primzahlordnung ℓ. Eine Paarung ist eine nicht-degenerierte bilineare Abbildung

$$e : G_1 \times G_2 \to G_T.$$

Eigenschaften:

- Paarungwerte $e(P,Q)$ sollen leicht ausgerechnet werden können.
- G_1, G_2, G_T sollen sicheres DLP und CDH haben. Aber für $G_1 = G_2$ ist das DDH in G_1 mit Hilfe von e leicht zu lösen! (Wie?)
- Meist G_1, G_2 Punktgruppe einer elliptischen Kurve, G_T Untergruppe von $\mathbb{F}_{q^k}^\times$ und e Weil- oder Tatepaarung.

Referenzprobleme:

1. Pairing inversion: Zu P, W das Element Q mit $e(P,Q) = W$ finden, etc.
2. Bilinear CDH: Zu P, aP, bP, cP den Wert $e(P,P)^{abc}$ berechnen.
3. viele mehr.

Paarungsbasierte Kryptographie

Destruktive Seite von Paarungen:

- MOV Angriff und FR Reduktion. (1991-1993)

Konstruktive Seite von Paarungen:

- Sakai, Ohgishi, Kasahara und Boneh, Franklin mit identitätsbasierter Kryptographie. (2000)

Seitdem extrem aktives Gebiet in der Kryptographie, mit vielen weiteren Anwendungen.

Forschung:

- Protokolle (Paarung abstrakt gegeben).
- Konstruktion geeigneter elliptischer Kurven, effiziente Berechung der Paarungen.

Paarungsbasierte Kryptographie

Beispiel: Short Signatures.

Setup wie Undeniable Signatures, plus Paarung $e : G \times G \to G_T$.

Schlüssel: $y = g^x$.
Signatur: (σ, M) mit $\sigma = H(M)^x$.
Verifikation: $e(g,\sigma) = e(y, H(M))$.

Gegenüber Schnorr spart man sich im wesentlichen h.

Die Sicherheit des Verfahrens kann im Zufallsorakelmodell auf das CDH in G zurückgeführt werden.

Identitätsbasierte Kryptographie

Traditionelle Verfahren:

- Private Keys werden geheim erzeugt, daraus die Public Keys berechnet. Bindung an eine Identität erfolgt durch eine Trust Authority mittels Zertifikat.

Identitätsbasierte Kryptographie:

- Hier werden erst die Public Keys erzeugt, und dann durch die TA daraus die Private Keys. Liefert Authentifizierung der Public Keys!
- Hat erweiterte Sicherheitsmodelle, Angreifer dürfen auch für andere Identitäten Fragen stellen, speziell sich geheime Schlüssel anderer Identitäten geben lassen.

Identitätsbasierte Verschlüsselung:

- Boneh und Franklin 2000. Es war bis dahin ungelöst, ob das überhaupt geht.

Identitätsbasierte Kryptographie

Identitätsbasierte Signaturen:

- Geht auch ohne Paarungen (Paper von Shamir in den 80igern, RSA basiert).
- Verwendet man $p(x) = e(P,x)$ in unserer Beschreibung des Schnorrverfahrens ($P = g$), dann erhält man ein identitätsbasiertes Signaturverfahren.

Beispiel:

P Erzeuger, sP öffentlicher Schlüssel von TA, s geheimer Schlüssel von TA. $H(ID)$ öffentlicher Schlüssel von ID, $sH(ID)$ geheimer Schlüssel von ID.

Signatur: $r = e(P,kP)$, $h = H(M||r)$, $u = hsH(ID) + kP$.

Verifikation: $y = e(sP,H(ID))$. Teste $e(P,u) = y^{H(M||r)}r$.