Identity based signatures

Ilko Müller

23. November 2005

Modell

- Alice signiert eine Nachricht mit ihrem geheimen Schlüssel

Modell

- Alice signiert eine Nachricht mit ihrem geheimen Schlüssel
- Bob verifiziert mit Alices öffentlichem Schlüssel

Modell

- Alice signiert eine Nachricht mit ihrem geheimen Schlüssel
- Bob verifiziert mit Alices öffentlichem Schlüssel
- vorher muss geklärt werden, dass der öffentliche Schlüssel zur Person Alice gehört

Modell

- Alice signiert eine Nachricht mit ihrem geheimen Schlüssel
- Bob verifiziert mit Alices öffentlichem Schlüssel
- vorher muss geklärt werden, dass der öffentliche Schlüssel zur Person Alice gehört
- mittels digitalem Zertifikat

ID-based signatures

- 1984 schrieb Shamir, dass es effizienter wäre, wenn diese Bindung nicht benötigt würde
- Identität des Benutzers $\longrightarrow$ öffentlicher Schlüssel

Angriffe

- Ziele des Angreifers:
  - existentielle Fälschung

Angriffe

- Ziele des Angreifers:
  - existentielle Fälschung
  - universelle Fälschung

Angriffe

- Ziele des Angreifers:
  - existentielle Fälschung
  - universelle Fälschung
  - total break

Fähigkeiten

- Fähigkeiten des Angreifers
  - key-only Angriff

Fähigkeiten

- Fähigkeiten des Angreifers
  - key-only Angriff
  - known-signature Angriff

Fähigkeiten

- Fähigkeiten des Angreifers
  - key-only Angriff
  - known-signature Angriff
  - chosen-message Angriff

Schnorr-Signatur

- Setup:
  - $p : (G, +) \rightarrow (V, \cdot)$ sei Einwegmonomorphismus und $ord(G) = l$ mit $l \in \mathbb{P}$ (z. Bsp.: $p : \mathbb{Z}\ /l\mathbb{Z} \rightarrow V, x \mapsto g^x$)
  - H sei ein Hashfkt. nach $\mathbb{Z}/l\mathbb{Z}$

Schnorr-Signatur

- Setup:
  - $p : (G, +) \rightarrow (V, \cdot)$ sei Einwegmonomorphismus und $ord(G) = l$ mit $l \in \mathbb{P}$ (z. Bsp.: $p : \mathbb{Z} / l\mathbb{Z} \rightarrow V, x \mapsto g^x$)
  - H sei ein Hashfkt. nach $\mathbb{Z}/l\mathbb{Z}$
- Schlüsselerzeugung:
  - sei $x \in G$ zufällig, $y := p(x)$ (z. Bsp.: $y = g^x$)
  - y öffentlicher Schlüssel, x geheimer Schlüssel

Schnorr-Signatur

- Setup:
  - $p : (G, +) \rightarrow (V, \cdot)$ sei Einwegmonomorphismus und $ord(G) = l$ mit $l \in \mathbb{P}$ (z. Bsp.: $p : \mathbb{Z} / l\mathbb{Z} \rightarrow V, x \mapsto g^x$)
  - H sei ein Hashfkt. nach $\mathbb{Z}/l\mathbb{Z}$
- Schlüsselerzeugung:
  - sei $x \in G$ zufällig, $y := p(x)$ (z. Bsp.: $y = g^x$)
  - y öffentlicher Schlüssel, x geheimer Schlüssel
- Sign(M):
  - sei $k \in G$ zufällig, $r := p(k)$ (z. Bsp.: $r = g^k$)
  - $h := H(M \parallel r)$, $u := hx + k$
  - Signatur ist dann: $\sigma(u, r)$

Schnorr-Signatur

- Setup:
  - $p : (G, +) \to (V, \cdot)$ sei Einwegmonomorphismus und $ord(G) = l$ mit $l \in \mathbb{P}$ (z. Bsp.: $p : \mathbb{Z}\ /l\mathbb{Z} \to V, x \mapsto g^x$)
  - H sei ein Hashfkt. nach $\mathbb{Z}/l\mathbb{Z}$
- Schlüsselerzeugung:
  - sei $x \in G$ zufällig, $y := p(x)$ (z. Bsp.: $y = g^x$)
  - y öffentlicher Schlüssel, x geheimer Schlüssel
- Sign(M):
  - sei $k \in G$ zufällig, $r := p(k)$ (z. Bsp.: $r = g^k$)
  - $h := H(M \parallel r)$, $u := hx + k$
  - Signatur ist dann: $\sigma(u, r)$
- Verify:
  - Signatur akzeptieren $\Leftrightarrow p(u) = y^{H(M\|r)} r$

Sicherheit der Schnorr-Signatur

- sicher im RO bzgl. existentieller Fälschung unter adaptive-chosen-message Angriff

Sicherheit der Schnorr-Signatur

- sicher im RO bzgl. existentieller Fälschung unter adaptive-chosen-message Angriff
- denn: erfolgreicher Angriff $\Rightarrow$ DLP lösbar

Schema von J.Cha und J.Cheon

Schema von J.Cha und J.Cheon

Computation Diffie-Hellman Problem

- setting: $(G, +)$ sei Gruppe mit $ord(G) = l$, $l \in \mathbb{P}$
- setting: sei P Erzeuger und $a, b, c \in \mathbb{Z}/l\mathbb{Z}$
- geg. sei (P, aP, bP)
- berechne abP

Decisional Diffie-Hellman Problem

- setting: wie oben
- geg. sei (P, aP, bP, cP)
- entscheide, ob $c = ab$ in $\mathbb{Z}/l\mathbb{Z}$
- falls ja: (P, aP, bP, cP) ist gültiges DH Tupel

Gap Diffie-Hellman Gruppe

- G heisst Gap Diffie-Hellman Gruppe, falls
  1. DDHP lösbar
  2. CDHP hart
- Bsp.: (hyper)elliptische Kurven mit Weil- oder Tate-pairing

Setup

- G sei Gruppe mit $ord(G) = l$, $l \in \mathbb{P}$

Setup

- G sei Gruppe mit $ord(G) = l$, $l \in \mathbb{P}$
- wähle Erzeuger P von G sowie $s \in \mathbb{Z}/l\mathbb{Z}$

Setup

- G sei Gruppe mit $ord(G) = l$, $l \in \mathbb{P}$
- wähle Erzeuger P von G sowie $s \in \mathbb{Z}/l\mathbb{Z}$
- setze $P_{Pub} := sP$

Setup

- G sei Gruppe mit $ord(G) = l$, $l \in \mathbb{P}$
- wähle Erzeuger P von G sowie $s \in \mathbb{Z}/l\mathbb{Z}$
- setze $P_{Pub} := sP$
- wähle Hashfkt. $H_1 : \{0,1\}^* \times G \rightarrow \mathbb{Z}/l\mathbb{Z}$ und $H_2 : \{0,1\}^* \rightarrow G$

Extract

- gegeben sei Identität *ID*

Extract

- gegeben sei Identität ID
- berechne $D_{ID} = sH_2(ID)$

Extract

- gegeben sei Identität ID
- berechne $D_{ID} = sH_2(ID)$
- D_{ID} privater Schlüssel

Extract

- gegeben sei Identität ID
- berechne $D_{ID} = sH_2(ID)$
- D_{ID} privater Schlüssel
- $Q_{ID} = H_2(ID)$ öffentlicher Schlüssel

Sign

- um Nachricht m zu signieren, wähle $r \in \mathbb{Z}/l\mathbb{Z}$ zufällig

Sign

- um Nachricht m zu signieren, wähle $r \in \mathbb{Z}/l\mathbb{Z}$ zufällig
- berechne $U = rQ_{ID}$, $h = H_1(m, U)$ und $V = (r + h)D_{ID}$

Sign

- um Nachricht m zu signieren, wähle $r \in \mathbb{Z}/l\mathbb{Z}$ zufällig
- berechne $U = rQ_{ID}$, $h = H_1(m, U)$ und $V = (r + h)D_{ID}$
- Signatur: $\sigma = (U, V)$

Verify

- geg. m, $\sigma = (U, V)$ und ID
- Signatur ok $\Leftrightarrow$ $(P, P_{Pub}, U + hQ_{ID}, V)$ ist gültiges DH Tupel

Konsistenz

- sei $\sigma(U, V)$ gültige Signatur für Nachricht m der Identität ID
  $\Rightarrow$
- $(P, P_{Pub}, U + hQ_{ID}, V) = (P, P_{Pub}, (r + h)Q_{ID}, V)$
  $= (P, sP, (r + h)Q_{ID}, s(r + h)Q_{ID})$

Orakel

- Identity Hash Oracle: gibt für jede ID den Hashwert $H_2(ID)$ aus

Orakel

- Identity Hash Oracle: gibt für jede ID den Hashwert $H_2(ID)$ aus
- Message Hash Oracle: gibt für jedes m und $U \in G$ den Hashwert $H_1(m, U)$ aus

Orakel

- Identity Hash Oracle: gibt für jede ID den Hashwert $H_2(ID)$ aus
- Message Hash Oracle: gibt für jedes m und $U \in G$ den Hashwert $H_1(m, U)$ aus
- Extraction Oracle: gibt für jede ID den den privaten Schlüssel D_{ID} aus

Orakel

- Identity Hash Oracle: gibt für jede ID den Hashwert $H_2(ID)$ aus
- Message Hash Oracle: gibt für jedes m und $U \in G$ den Hashwert $H_1(m, U)$ aus
- Extraction Oracle: gibt für jede ID den den privaten Schlüssel D_{ID} aus
- Signature Oracle: gibt für jede ID und m die Signatur aus

Theorem

Theorem: Seien q_{H_1}, q_{H_2}, q_S und q_E die maximale Anzahl von Anfragen an die Orakel. Wenn es einen Algorithmus A_0 für einen adaptive-chosen-message und *ID* Angriff für dieses Schema gibt, mit Laufzeit t_0 und Vorteil $\epsilon_0 \geq \frac{10(q_S+1)(q_S+q_{H_1})q_{H_2}}{(l-1)}$, dann kann CDHP in der Zeit $t \leq \frac{120686 q_{H_1} q_{H_2} t_0}{\epsilon_0(1-\frac{1}{l})}$ gelöst werden.

Fazit

- Schema ist sicher, wenn CDHP schwer ist, bzw. wenn G eine Gap Diffie-Hellman Gruppe ist

Zusammenhangmit BF-IDE

- das Setup des ID Encryption Schemas kann für dieses Schema genutzt werden

Schema von Prof. Hess

Schema von Prof. Hess

Vorraussetzungen

- $(G, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l

Vorraussetzungen

- $(G, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l
- $P \in G$ sei Erzeuger von G

Vorraussetzungen

- $(G, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l
- $P \in G$ sei Erzeuger von G
- $e : G \times G \longrightarrow V$ sei pairing mit folgenden Eigenschaften:
  1. Bilinear: $e(x_1 + x_2, y) = e(x_1, y)e(x_2, y)$ und $e(x, y_1 + y_2) = e(x, y_1)e(x, y_2)$
  2. nicht-degeneriert: $\exists x, y \in G$ so dass $e(x, y) \neq 1$

Vorraussetzungen

- $(G, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l
- $P \in G$ sei Erzeuger von G
- $e : G \times G \longrightarrow V$ sei pairing mit folgenden Eigenschaften:
  1. Bilinear: $e(x_1 + x_2, y) = e(x_1, y)e(x_2, y)$ und $e(x, y_1 + y_2) = e(x, y_1)e(x, y_2)$
  2. nicht-degeneriert: $\exists x, y \in G$ so dass $e(x, y) \neq 1$
- Hashfunktionen $h : \{0,1\}^* \times V \longrightarrow (\mathbb{Z}/l\mathbb{Z})^\times$, $H : \{0,1\}^* \longrightarrow G^*$

Setup

- trust authority (TA) wählt zufälliges $t \in (\mathbb{Z}/l\mathbb{Z})^{\times}$

Setup

- trust authority (TA) wählt zufälliges $t \in (\mathbb{Z}/l\mathbb{Z})^{\times}$
- TA berechnet $Q_{TA} = tP$

Setup

- trust authority (TA) wählt zufälliges $t \in (\mathbb{Z}/l\mathbb{Z})^{\times}$
- TA berechnet $Q_{TA} = tP$
- Q_{TA} öffentlich, t ist master secret

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $S_{ID} = tH(ID)$

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $S_{ID} = tH(ID)$
- wird normalerweise einmal pro ID gemacht

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $S_{ID} = tH(ID)$
- wird normalerweise einmal pro ID gemacht
- TA benutzt die gleichen Setup-Daten für viele verschiedene IDs

Sign

- um Nachricht m zu signieren, wähle beliebiges $P_1 \in G^*$ und zufälliges $k \in (\mathbb{Z}/l\mathbb{Z})^\times$

Sign

- um Nachricht m zu signieren, wähle beliebiges $P_1 \in G^*$ und zufälliges $k \in (\mathbb{Z}/l\mathbb{Z})^\times$
- berechne:
  1. $r = e(P_1, P)^k$
  2. $v = h(m, r)$
  3. $u = vS_{ID} + kP_1$

Sign

- um Nachricht m zu signieren, wähle beliebiges $P_1 \in G^*$ und zufälliges $k \in (\mathbb{Z}/I\mathbb{Z})^\times$
- berechne:
  1. $r = e(P_1, P)^k$
  2. $v = h(m, r)$
  3. $u = vS_{ID} + kP_1$
- die Signatur ist dann $(u, v) \in (G, (\mathbb{Z}/I\mathbb{Z})^\times)$

Verify

- geg. Nachricht m, Signatur (u, v)

Verify

- geg. Nachricht m, Signatur (u, v)
- berechne:
  1. $r = e(u, P)e(H(ID), -Q_{TA})^v$
  2. Signatur ok $\Leftrightarrow v = h(m, r)$

Konsistenz

- für gültige Signatur (u, v) gilt:
  $r = e(u, P)e(H(ID), -Q_{TA})^v \Leftrightarrow$
- $e(P_1, P)^k = e(vtH(ID) + kP_1, P) \cdot e(H(ID), -tP)^v$
  $= e(vtH(ID), P) \cdot e(kP_1, P) \cdot e(H(ID), -tP)^v$
  $= e(vtH(ID), P) \cdot e(-vtH(ID), P) \cdot e(P_1, P)^k$
  $= e(0, P) \cdot e(P_1, P)^k$

Sicherheit

Theorem: Wenn im RO Modell ein Angreifer A existiert, der höchstens $n_1 \geq 1$ Anfragen an die Identity Hash und Extraction Oracle, sowie höchstens $n_2 \geq 1$ Anfragen an die Message Hash und Signature Oracle macht und innerhalb der Zeit T_A eine existentielle Fälschung mit W'keit $\epsilon_A \geq \frac{an_1 n_2^2}{l}$ für eine Konstante $a \in \mathbb{Z}^{\geq 1}$ erzeugt, dann existiert ein anderer Algorithmus C und eine Konstante $c \in \mathbb{Z}^{\geq 1}$, so dass C das DHP bzgl. (P, Q_{TA}, R) für beliebige $R \in G^*$ in der Zeit $T_C \leq \frac{cn_1 n_2 T_A}{\epsilon_A}$ löst.

Vergleich

	Hess Schema	CC Schema
Signing	1E+1M	2M
Verifying	1E+2P / 1E+1P	1M+2P
Signature	$G \times (\mathbb{Z}/l\mathbb{Z})^{\times}$	$G \times G$

Tabelle: Effizienzvergleich

- E einmaliges Exponieren in V
- M skalare Multiplikation in G
- P Berechnung eines pairings

Problem

- Problem: TA kennt geheimen Schlüssel einer Person

Problem

- Problem: TA kennt geheimen Schlüssel einer Person
- TA kann Nachrichten signieren, als ob sie von dieser Person stammt

Problem

- Problem: TA kennt geheimen Schlüssel einer Person
- TA kann Nachrichten signieren, als ob sie von dieser Person stammt
- u.U. für entsprechendes Encryption Schema durchaus gewollt

key escrow

- es gebe die trust authorities TA_i für $1 \leq i \leq n$

key escrow

- es gebe die trust authorities TA_i für $1 \leq i \leq n$
- master secret t wird auf auf die TA_i wie folgt verteilt:
  - jede TA_i erzeugt eigenen privaten Schlüssel t_i unabhängig voneinander und veröffentlicht $Q_{TA_i} = t_i P$
  - master private key ist dann def. als $t = \sum_{i=1}^{n} t_i$

key escrow

- es gebe die trust authorities TA_i für $1 \leq i \leq n$
- master secret t wird auf auf die TA_i wie folgt verteilt:
  - jede TA_i erzeugt eigenen privaten Schlüssel t_i unabhängig voneinander und veröffentlicht $Q_{TA_i} = t_i P$
  - master private key ist dann def. als $t = \sum_{i=1}^{n} t_i$
- master public key ist dann $Q_{TA} = \sum_{i=1}^{n} Q_{TA_i}$

key escrow

- es gebe die trust authorities TA_i für $1 \leq i \leq n$
- master secret t wird auf auf die TA_i wie folgt verteilt:
  - jede TA_i erzeugt eigenen privaten Schlüssel t_i unabhängig voneinander und veröffentlicht $Q_{TA_i} = t_i P$
  - master private key ist dann def. als $t = \sum_{i=1}^{n} t_i$
- master public key ist dann $Q_{TA} = \sum_{i=1}^{n} Q_{TA_i}$
- Signierer erhält Teil seines privaten Schlüssels via $S_{ID}^{(i)} = t_i Q_{ID}$

key escrow

- es gebe die trust authorities TA_i für $1 \leq i \leq n$
- master secret t wird auf auf die TA_i wie folgt verteilt:
  - jede TA_i erzeugt eigenen privaten Schlüssel t_i unabhängig voneinander und veröffentlicht $Q_{TA_i} = t_i P$
  - master private key ist dann def. als $t = \sum_{i=1}^{n} t_i$
- master public key ist dann $Q_{TA} = \sum_{i=1}^{n} Q_{TA_i}$
- Signierer erhält Teil seines privaten Schlüssels via $S_{ID}^{(i)} = t_i Q_{ID}$
- privater Schlüssel des Signierers ist dann $S_{ID} = \sum_{i=1}^{n} S_{ID}^{(i)}$

key escrow

- eine oder mehrere TA_i geben falsches $S_{ID}^{(i)}$ bzgl. Q_{TA_i} an den Signierer

key escrow

- eine oder mehrere TA_i geben falsches $S_{ID}^{(i)}$ bzgl. Q_{TA_i} an den Signierer
- erzeugt ungültigen privaten Schlüssel

key escrow

- eine oder mehrere TA_i geben falsches $S_{ID}^{(i)}$ bzgl. Q_{TA_i} an den Signierer
- erzeugt ungültigen privaten Schlüssel
- Signierer kann Daten der einzelnen TA_i überprüfen, indem er eine Nachricht nur mit den Werten der einzelnen TA_i signiert und verifiziert

Schema von Prof. Hess (allgemeine Version)

Schema von Prof. Hess (allgemeine Version)

Vorraussetzungen

- $(G, +)$, $(G_1, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l

Vorraussetzungen

- $(G, +)$, $(G_1, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l
- $p : G \longrightarrow V$ sei effizient berechenbarer Monomorphismus

Vorraussetzungen

- $(G, +)$, $(G_1, +)$ und $(V, \cdot)$ seien zyklische Gruppen mit primer Ordnung l
- $p : G \rightarrow V$ sei effizient berechenbarer Monomorphismus
- Hashfunktionen $h : \{0,1\}^* \times V \rightarrow (\mathbb{Z}/l\mathbb{Z})^\times \times (\mathbb{Z}/l\mathbb{Z})^\times$ sowie $H : \{0,1\}^* \rightarrow G_1^*$

Setup

- *TA* wählt effizient berechenbare Monomorphismen $s : G_1 \longrightarrow G$ und $q : G_1 \longrightarrow V$ mit $p(s(x)) = q(x)\ \forall x \in G_1$

Setup

- *TA* wählt effizient berechenbare Monomorphismen $s : G_1 \longrightarrow G$ und $q : G_1 \longrightarrow V$ mit $p(s(x)) = q(x)\ \forall x \in G_1$
- q öffentlich

Setup

- *TA* wählt effizient berechenbare Monomorphismen $s : G_1 \longrightarrow G$ und $q : G_1 \longrightarrow V$ mit $p(s(x)) = q(x)\ \forall x \in G_1$
- q öffentlich
- s bleibt geheim

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $a = s(H(ID))$

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $a = s(H(ID))$
- a geheimer Schlüssel

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet TA $a = s(H(ID))$
- a geheimer Schlüssel
- $y = q(H(ID))$ öffentlicher Schlüssel

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet *TA* $a = s(H(ID))$
- a geheimer Schlüssel
- $y = q(H(ID))$ öffentlicher Schlüssel
- wird normalerweise einmal pro ID gemacht

Extract

- wenn Signierer seinen geheimen Schlüssel will, berechnet *TA* $a = s(H(ID))$
- a geheimer Schlüssel
- $y = q(H(ID))$ öffentlicher Schlüssel
- wird normalerweise einmal pro ID gemacht
- *TA* benutzt die gleichen Setup-Daten für viele verschiedene IDs

Sign(m)

- um m zu signieren, wählt Signierer zufälliges $k \in G^*$ und berechnet:
  1. $r = p(k)$
  2. $(v, w) = h(m, r)$
  3. $u = va + wk$

Sign(m)

- um m zu signieren, wählt Signierer zufälliges $k \in G^*$ und berechnet:
  1. $r = p(k)$
  2. $(v, w) = h(m, r)$
  3. $u = va + wk$
- Signatur: $(u, r) \in G \times V^*$

Verify

- geg. Nachricht m und Signatur (u, r) berechne:
  1. $(v, w) = h(m, r)$
  2. $y = q(H(ID))$

Verify

- geg. Nachricht m und Signatur (u, r) berechne:
  1. $(v, w) = h(m, r)$
  2. $y = q(H(ID))$
- Signatur ok $\Leftrightarrow p(u) = y^v r^w$

Konsistenz

- für gültige Signatur (u, r) gilt:

Konsistenz

- für gültige Signatur (u, r) gilt:
- $p(u) = p(va + wk)$
  $= p(a)^v p(k)^w$
  $= p(s(H(ID)))^v p(k)^w$
  $= q(H(ID))^v p(k)^w$
  $= y^v r^w$

Sicherheit

Theorem: Wenn im RO Modell ein Angreifer A existiert, der höchstens $n_1 \geq 1$ Anfragen an die Identity Hash und Extraction Oracle, sowie höchstens $n_2 \geq 1$ Anfragen an die Message Hash und Signature Oracle macht und innerhalb der Zeit T_A eine existentielle Fälschung mit W'keit $\epsilon_A \geq \frac{an_1 n_2^2}{l}$ für eine Konstante $a \in \mathbb{Z}^{\geq 1}$ erzeugt, dann existiert ein anderer Algorithmus C und eine Konstante $c \in \mathbb{Z}^{\geq 1}$, so dass C für beliebige Eingaben $Q = s(P)$ $P \in G_1^*, Q \in G^*$ $s(R)$ berechnet für beliebige $R \in G_1^*$ in erwarteter Zeit $T_C \leq \frac{cn_1 n_2 T_A}{\epsilon_A}$.

Verbindung zur Schnorr-Signatur

- sei $h_1 : \{0,1\}^* \times V \to (\mathbb{Z}/l\mathbb{Z})^\times$ und def. $h(m,r) := (h_1(m,r), 1)$

Verbindung zur Schnorr-Signatur

- sei $h_1 : \{0,1\}^* \times V \rightarrow (\mathbb{Z}/l\mathbb{Z})^\times$ und def. $h(m,r) := (h_1(m,r), 1)$
- $\Rightarrow w = 1$ und Gleichung der Verifizierung ist $p(u) = y^v r$

Verbindung zur Schnorr-Signatur

- sei $h_1 : \{0,1\}^* \times V \rightarrow (\mathbb{Z}/l\mathbb{Z})^\times$ und def. $h(m,r) := (h_1(m,r), 1)$
- $\Rightarrow w = 1$ und Gleichung der Verifizierung ist $p(u) = y^v r$
- Signatur $(u,v) \Leftrightarrow$ Signatur (u,r)

Verbindung zur Schnorr-Signatur

- sei $h_1 : \{0,1\}^* \times V \to (\mathbb{Z}/l\mathbb{Z})^\times$ und def. $h(m,r) := (h_1(m,r),1)$
- $\Rightarrow w = 1$ und Gleichung der Verifizierung ist $p(u) = y^v r$
- Signatur $(u,v) \Leftrightarrow$ Signatur (u,r)
- mit entsprechender Modifikation des Signing und des Verifying ergibt sich allgemeine Version der Schnorr-Signatur

Verbindung zum 1. Hess-Schema

- hier ist $(G_1, +) = (G, +)$

Verbindung zum 1. Hess-Schema

- hier ist $(G_1, +) = (G, +)$
- def.: $p(x) := e(x, P)$, $q(x) := e(x, Q_{TA})$ und $s(x) := tx$

Verbindung zum 1. Hess-Schema

- hier ist $(G_1, +) = (G, +)$
- def.: $p(x) := e(x, P)$, $q(x) := e(x, Q_{TA})$ und $s(x) := tx$
- P,Q_{TA} öffentlich also auch $p(x)$,$q(x)$

Verbindung zum 1. Hess-Schema

- hier ist $(G_1, +) = (G, +)$
- def.: $p(x) := e(x, P)$, $q(x) := e(x, Q_{TA})$ und $s(x) := tx$
- P,Q_{TA} öffentlich also auch $p(x)$,$q(x)$
- t nur der TA bekannt also $s(x)$ auch nur von TA berechenbar

Verbindung zum 1. Hess-Schema

- hier ist $(G_1, +) = (G, +)$
- def.: $p(x) := e(x, P)$, $q(x) := e(x, Q_{TA})$ und $s(x) := tx$
- P,Q_{TA} öffentlich also auch $p(x)$,$q(x)$
- t nur der TA bekannt also $s(x)$ auch nur von TA berechenbar
- mit $h(m, r) := (h_1(m, r), 1)$ (analog zu Schnorr-Signatur) ergibt sich das zuerst vorgestellte Hess'sche Schema

übliche public-key Signaturen

- sei $(G, +) = (V, \cdot)$ und q die Identität

übliche public-key Signaturen

- sei $(G, +) = (V, \cdot)$ und q die Identität
- TA ist identisch mit Signierer

übliche public-key Signaturen

- sei $(G, +) = (V, \cdot)$ und q die Identität
- TA ist identisch mit Signierer
- Signierer wählt zufälliges $a \in G^*$ und berechnet $y = p(a)$

übliche public-key Signaturen

- sei $(G, +) = (V, \cdot)$ und q die Identität
- TA ist identisch mit Signierer
- Signierer wählt zufälliges $a \in G^*$ und berechnet $y = p(a)$
- s ist Inverse zu p

übliche public-key Signaturen

- sei $(G, +) = (V, \cdot)$ und q die Identität
- TA ist identisch mit Signierer
- Signierer wählt zufälliges $a \in G^*$ und berechnet $y = p(a)$
- s ist Inverse zu p
- Bsp.: $p(x) = g^x$, s ist der diskrete Logarithmus bzgl. g