Bundesamt
für Sicherheit in der
Informationstechnik

Für jedes Problem das richtige Werkzeug: Stochastik bei Seitenkanalangriffen

Werner Schindler

Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn

Bonn, den 09. November 2007

Übersicht (I)

- Einführung
- Was sind Seitenkanalangriffe?
- Ausgangspunkt: Quisquaters Laufzeitattacke (Cardis 1998)
- Ausflug in die Stochastik
  - statistische Entscheidungstheorie
  - Analyse von Montgomerys Multiplikationsalgorithmus
  - modulare Exponentiation: Modellierung des Zeitverhaltens durch einen stochastischen Prozess

Übersicht (II)

- Quisquaters Laufzeitattacke: Effizienzsteigerung durch Stochastik
- Weitere Beispiele
  - Laufzeitangriffe gegen RSA mit CRT
  - Laufzeitangriffe gegen OpenSSL
  - kombinierte Laufzeit- und Powerangriffe
  - Cache-Attacke gegen Blockchiffren
  - Instructions-Cache-Attacke gegen OpenSSL
- Beobachtungen und Fazit

Mathematische Methoden in der „klassischen“ Kryptographie (I)

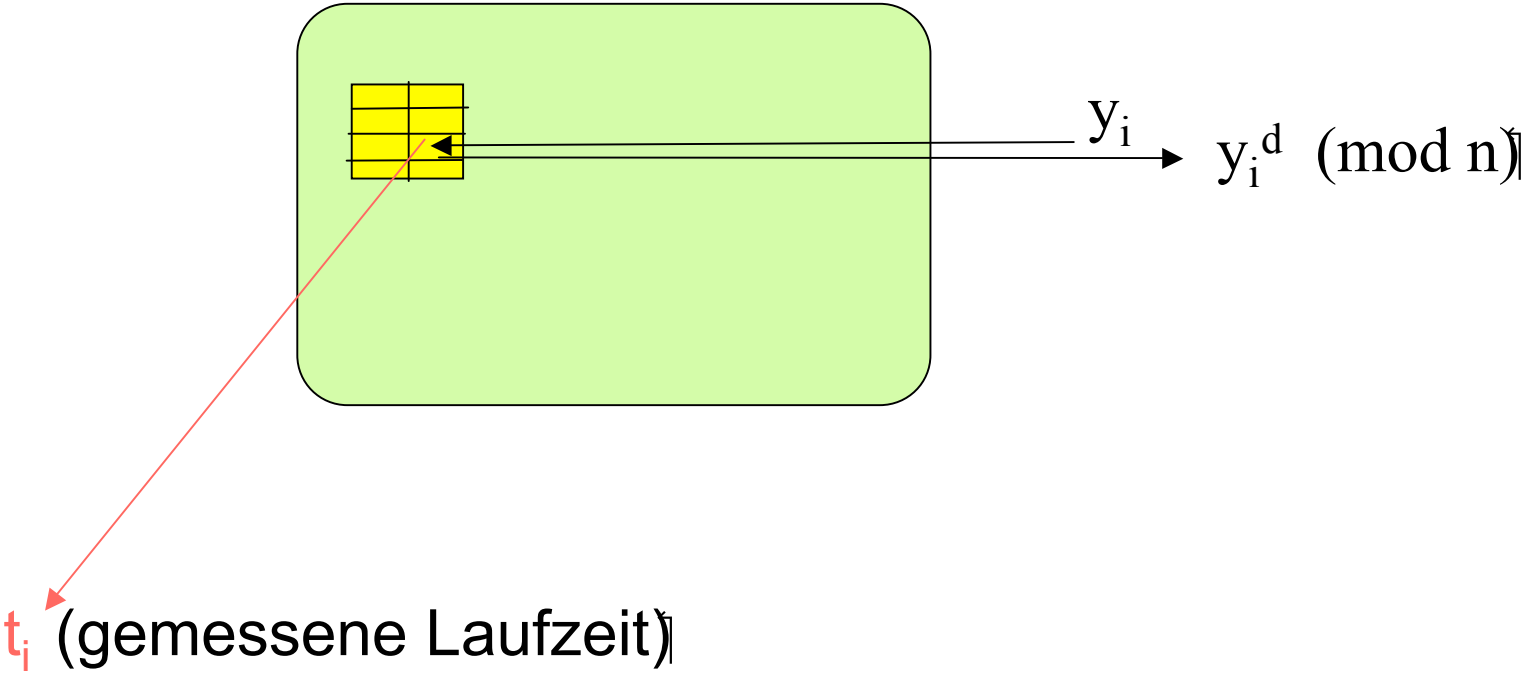
- Zahlentheorie (public key-Algorithmen)
- Algebra (Lösen nichtlinearer GL-Systeme, Codierungstheorie, HFE, ...)
- komplexitätstheoretische Ansätze (beweisbare Sicherheit, Effizienzbetrachtungen, ...)
- diskrete Fouriertransformation (Differentialattacke gegen Blockchiffren etc.)
- ad-hoc-Ansätze
- ...

Mathematische Methoden in der „klassischen“ Kryptographie (II)

Stochastik und Statistik spielen dagegen nur eine Nebenrolle

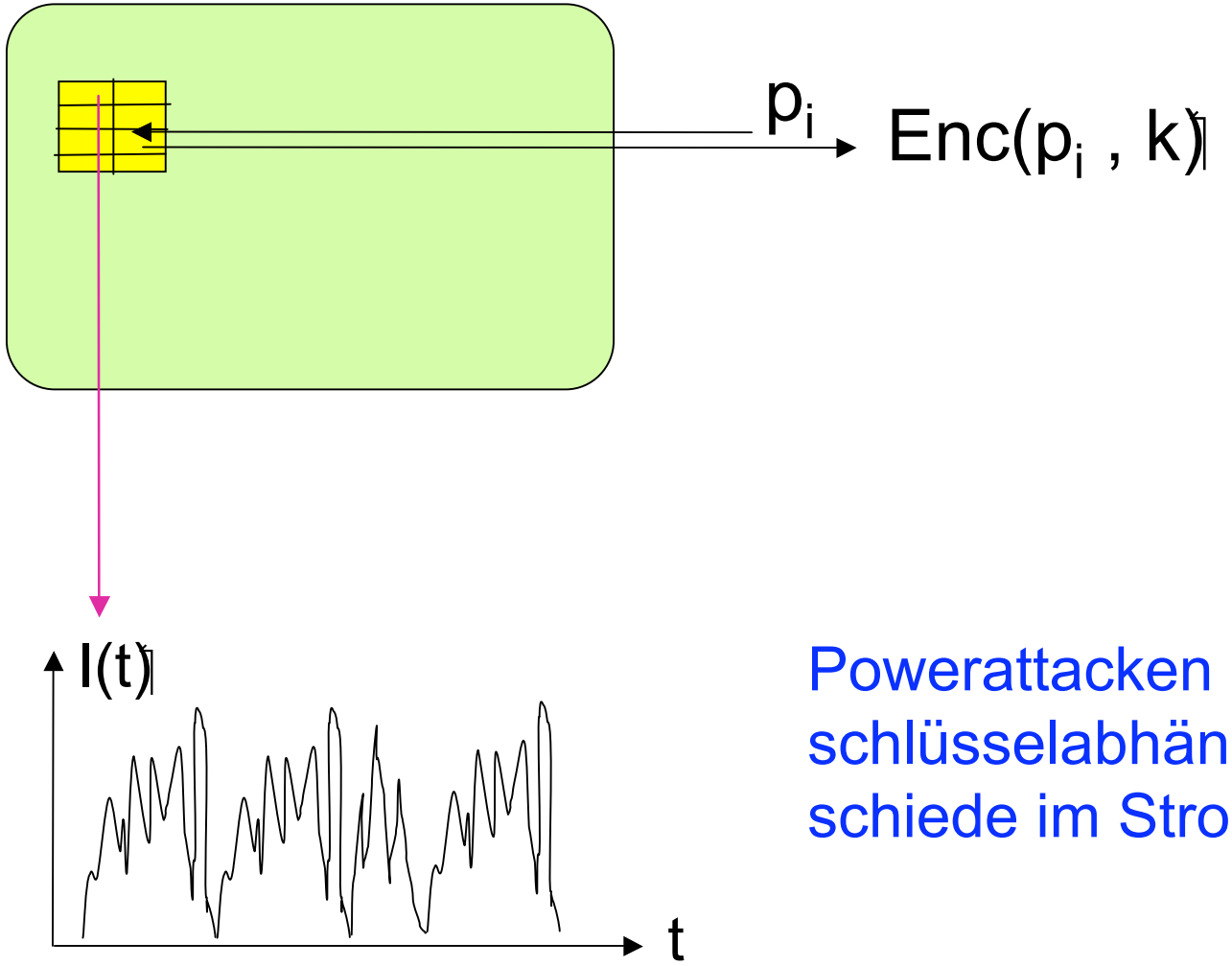
(Korrelationsangriffe gegen Stromchiffren, Time-Memory-Trade-offs, ...).

Laufzeitangriffe – Grundidee



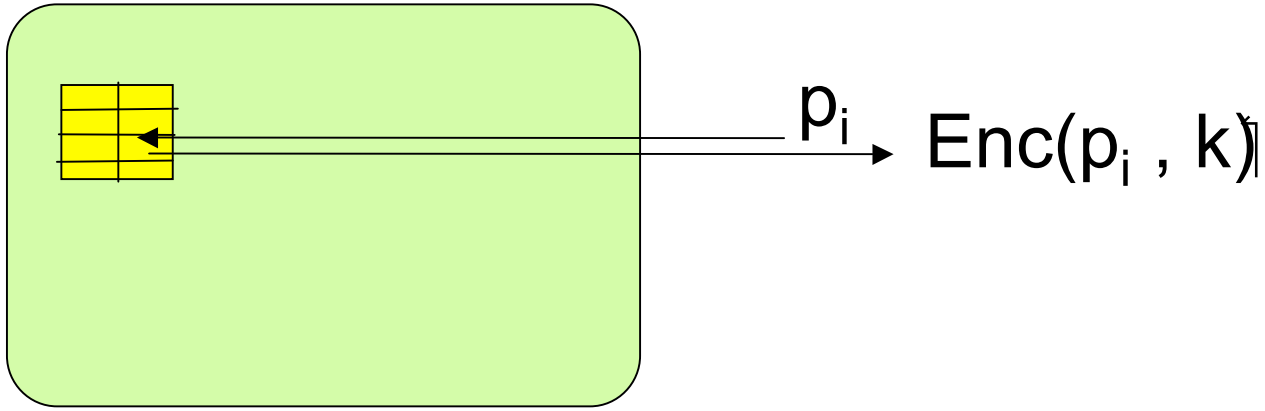
Laufzeitattacken nutzen schlüsselabhängige Laufzeitunterschiede.

Powerattacken – Grundidee



Powerattacken nutzen schlüsselabhängige Unterschiede im Stromverbrauch.

Abstrahlangriffe – Grundidee



Abstrahlangriffe nutzen die elektromagnetische Abstrahlung, um Aktivitäten auf dem Chip zu lokalisieren (→ Schlüssel).

- Bei einem Seitenkanalangriff „beobachtet“ der Angreifer ein Kryptosystem (Chipkarte, Software) „im laufenden Betrieb“ (Verschlüsselung, Signaturerstellung etc.). D.h., er führt physikalische Messungen durch, aber er beeinflusst das System nicht.

- Anmerkung: Dieser Vortrag konzentriert sich im Folgenden auf reine Laufzeitangriffe und kombinierte (= lokale) Laufzeitangriffe.

(Persönlicher) Ausgangspunkt

- Angriffe gegen RSA ohne CRT (Chinesischer Restsatz)
  - Kocher (1996): glänzende Idee, praktisch erprobt, aber theoretische Argumentation ziemlich fehlerhaft
  - Quisquater et al. (1998): anderer (komplizierterer) Multiplikationsalgorithmus, praktisch erprobt, heuristischer Ansatz

Persönlicher Eindruck: Geht das nicht besser?

Beispiel 1: Laufzeitangriff gegen RSA ohne CRT

Grundvoraussetzung:

- Gesamtlaufzeit t_j := Time (y_j^d (mod n)) hängt von Basis y_j ab (d,n fest)

Annahmen:

- Angreifer kann Time(a*b (mod n)) für beliebige a,b bestimmen
- hier: modulare Exponentiation mit square & multiply-Algorithmus.

Angriffsziel: geheimer Exponent d

Exponentiationsalgorithmus: square & multiply

berechnet $y \rightarrow y^d \pmod n$ mit $d = (d_{w-1},\ldots,d_0)_2$

temp := y

for i=w-2 down to 0 do {

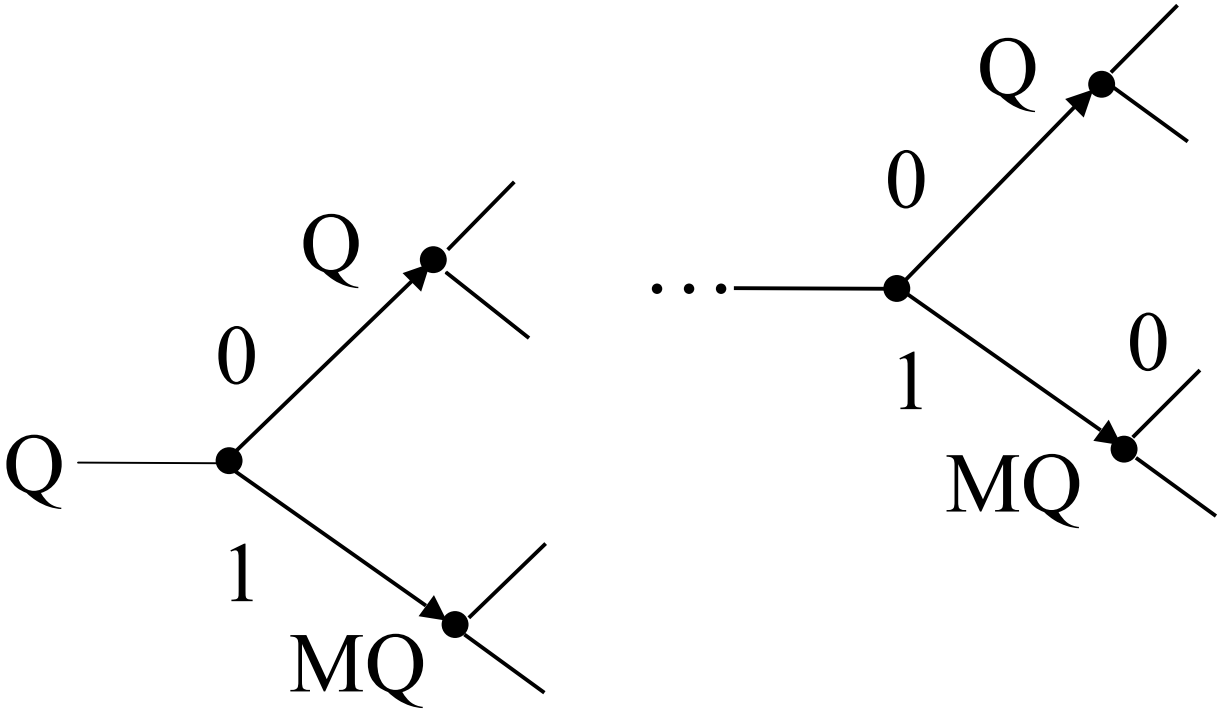
temp := $temp^2 \pmod n$

if (d_i = 1) then temp := temp * y (mod n)

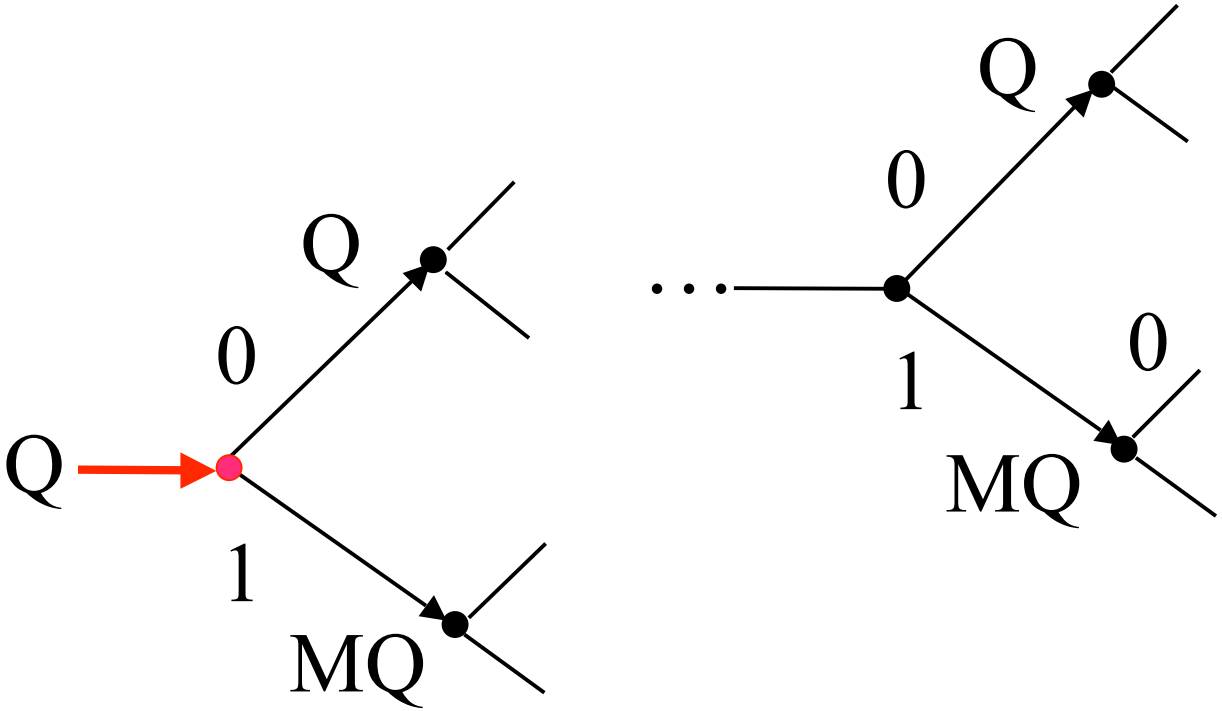
}

return temp (= $y^d \pmod n$)

$d_{w-1} = 1 \quad d_{w-2} \quad d_k \quad d_0$



$d_{w-1} = 1$ d_{w-2} d_k d_0



Exponentiationsalgorithmus: square & multiply

berechnet $y \rightarrow y^d \pmod n$ mit $d = (d_{w-1},\ldots,d_0)_2$

temp := y

for i=w-2 down to 0 do {

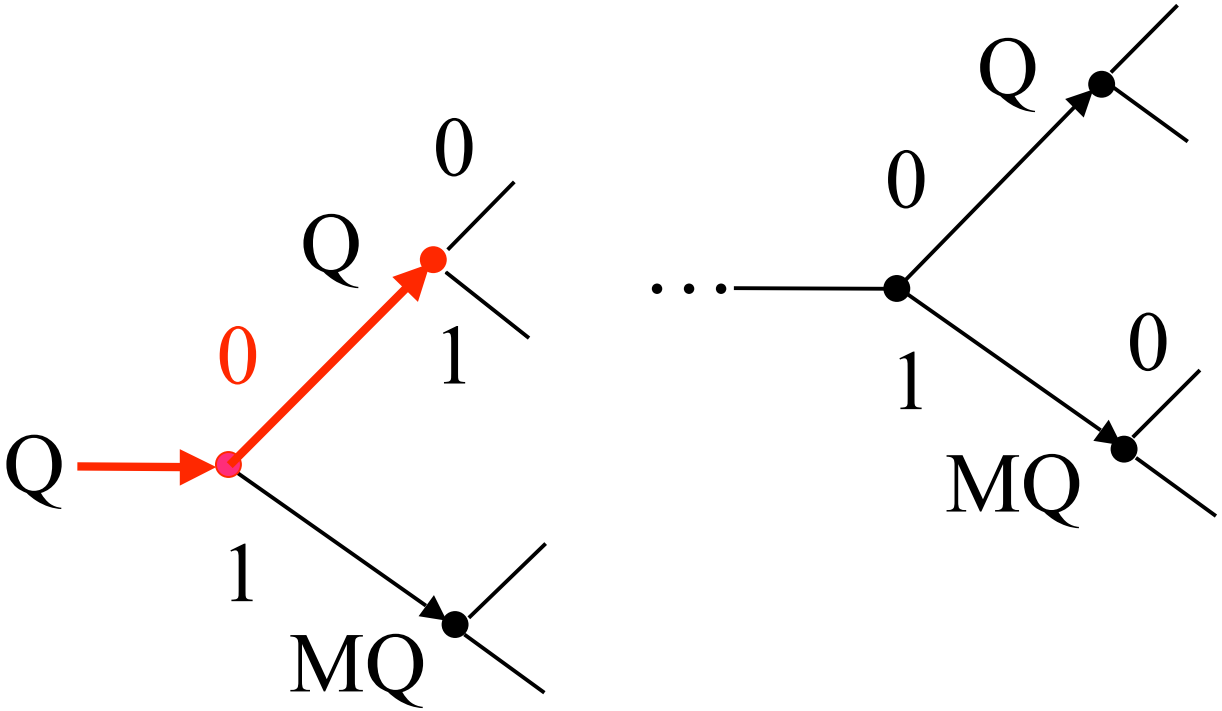
temp := $temp^2 \pmod n$

if (d_i = 1) then temp := temp * y (mod n)

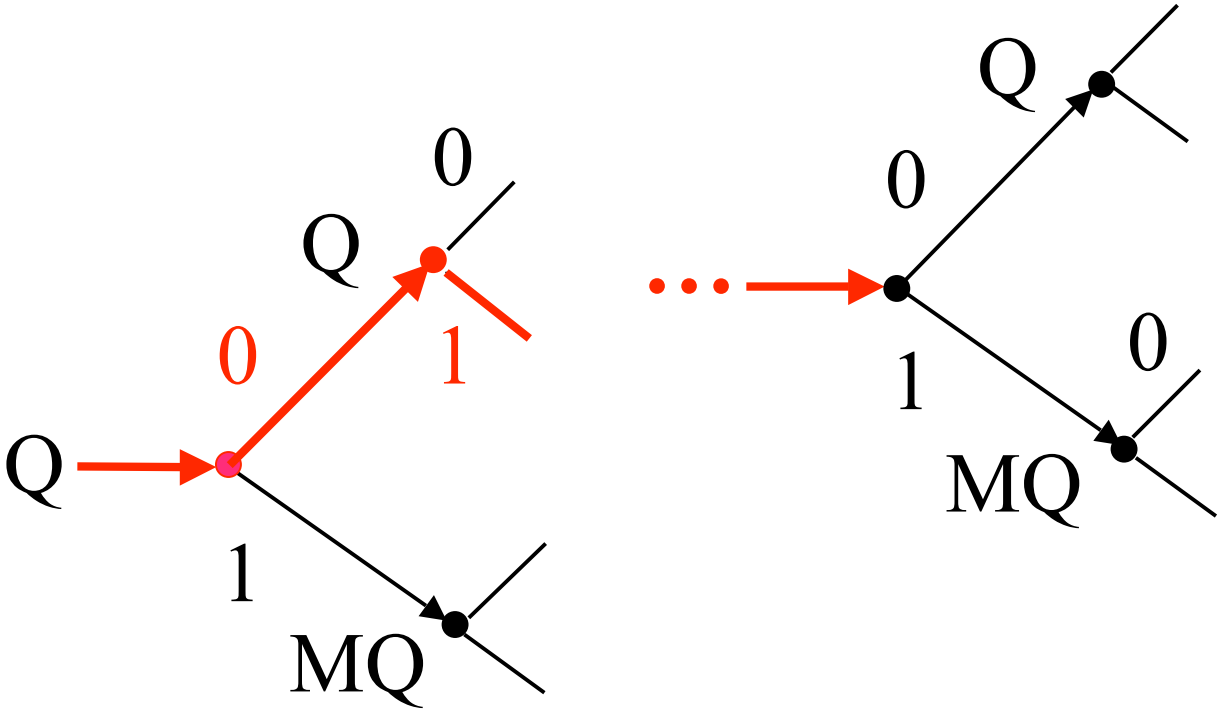
}

return temp (= $y^d \pmod n$)

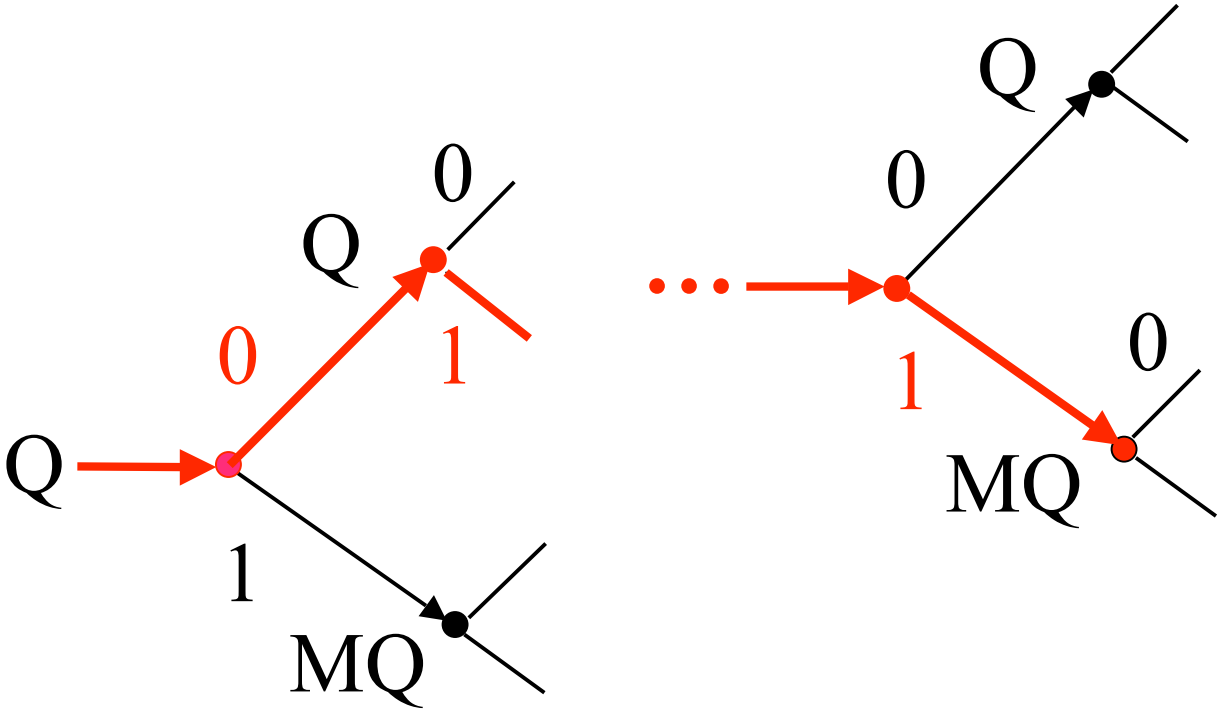
$d_{w-1} = 1$ d_{w-2} d_k d_0



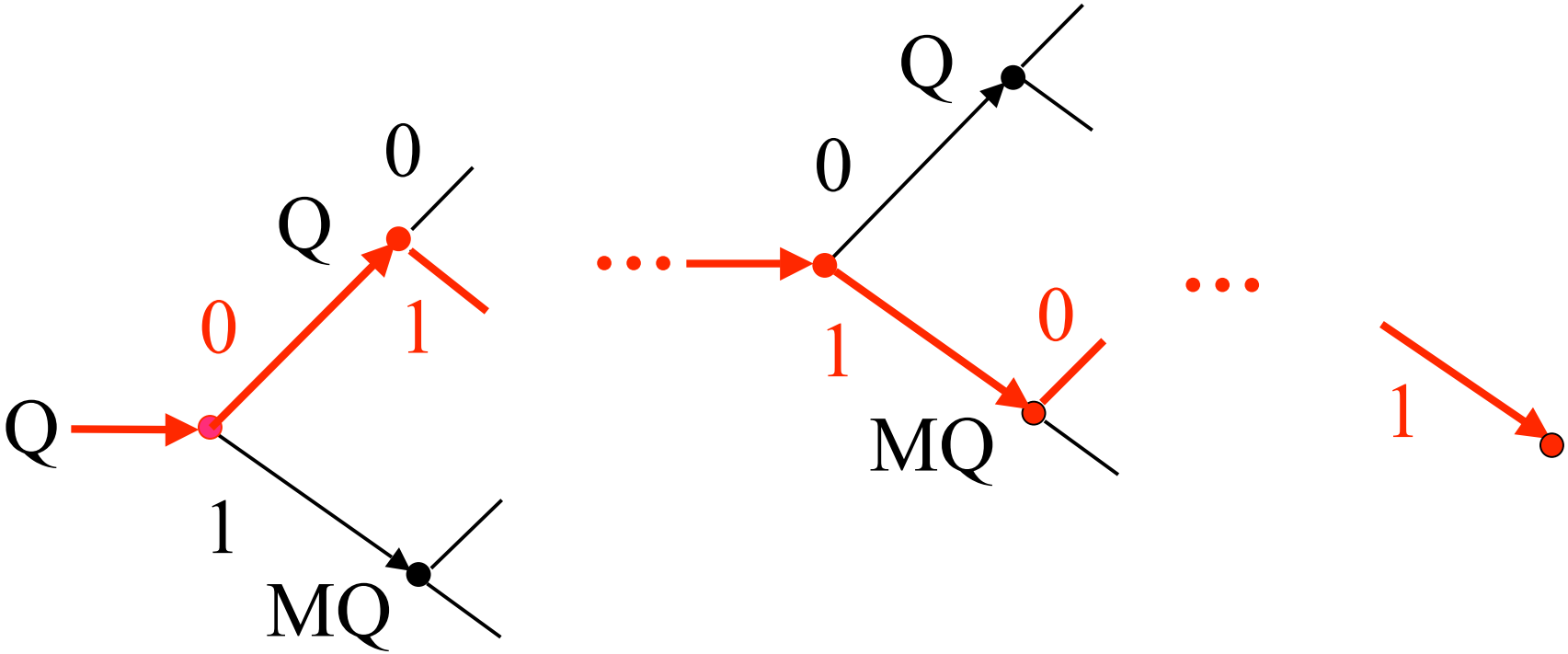
$d_{w-1} = 1$ d_{w-2} d_k d_0



$d_{w-1} = 1 \quad d_{w-2} \quad d_k \quad d_0$



$d_{w-1} = 1 \quad d_{w-2} \quad d_k \quad d_0$



Der richtige Pfad liefert d.

Vorgehen / Zielsetzung

- Interpretiere das Raten der Exponentenbits als eine Folge *statistischer Entscheidungsprobleme*.
- Wende stets die *optimale Entscheidungsstrategie an*.
- Finde eine *effiziente Fehlerkorrekturstrategie*.

Statistische Entscheidungstheorie (I)

<u>Ausgangssituation:</u> Gegeben ist eine Menge zulässiger Hypothesen $\Theta = \{\theta_0, \theta_1, \ldots, \theta_{k-1}\}$

hier: $\Theta = \{\theta_0, \theta_1\}$ (Erraten von d_k)
θ_i bedeutet, dass $d_k = i$

<u>Ziel:</u> Der Statistiker (hier: Angreifer / Designer) entscheidet sich auf Basis einer Beobachtung ω (hier: Laufzeiten) für eine Hypothese $\theta^* \in \Theta$.

Statistische Entscheidungstheorie (II)

Beobachtungen (hier: Laufzeiten) werden als Realisierungen von Zufallsvariablen interpretiert. Die Verteilung dieser Zufallsvariablen hängt von der Hypothese θ ab.

Die optimale Entscheidungsstrategie berücksichtigt

- die *Verteilung der Zufallsvariablen* für alle zulässigen Hypothesen $\theta_i \in \Theta$ (repräsentiert durch Wahrscheinlichkeitsdichten $f_\theta(\cdot)$)
  - = Hauptanteil an Entscheidungsstrategie
  - Berechnen der $f_\theta(\cdot)$: praktische Hauptschwierigkeit
- *a priori-Wissen:* Wie wahrscheinlich sind die einzelnen Hypothesen?
- Folgen fehlerhafter Entscheidungen (hier: *Aufwand, falsche Schätzer zu korrigieren*)

Unser Laufzeitangriff

<u>gesucht:</u>

- Exponentenbit $d_k \in \{0,1\} = \Theta$

- N = Stichprobenumfang
- Beobachtungsvektor: $(t_i, u_i, v_i, q_i)_{i \leq N}$,
  - t_i : Restlaufzeit für Bits k, ..., 0 (← **gemessene Gesamtlaufzeit** für Basis y_i , **Bitschätzer** für $d_{w-1},\ldots,d_{k+1}$)
  - u_i : Laufzeit für nächste Mult., falls $d_k=1$
  - v_i : Laufzeit für nächste Quad., falls $d_k=1$
  - q_i : Laufzeit für nächste Quad., falls $d_k=0$

Hauptschwierigkeit

Bestimme die bedingten W-Dichten

$f_\theta\ ((t_i \mid u_i, v_i, q_i)_{\ i \le N}\ )$ für $\theta = 0,1$

- Interpretiere die Zeiten für modulare Operationen *nach der nächsten Verzweigung* als Realisierungen von (geeignet definierten) Zufallsvariablen $W_{i,1}, W_{i,2}, \ldots$.

Exponentiationsalgorithmus: square & multiply

berechnet $y \rightarrow y^d \pmod n$ mit $d = (d_{w-1},\ldots,d_0)_2$

temp := y

for i=w-2 down to 0 do {

temp := $temp^2 \pmod n$

if (d_i = 1) then temp := temp * y (mod n)

}

return temp (= $y^d \pmod n$)

Stochastisches Modell (I)

Basis y_i:

Interpretiere die normierten Zwischenwerte

$$\frac{temp_{i,0}}{n}, \quad \frac{temp_{i,1}}{n}, \quad \frac{temp_{i,2}}{n}, \ldots$$

des s&m-Algorithmus als Realisierungen von [0,1)-wertigen Zufallsvariablen $S_{i,0}, S_{i,1}, \ldots$

- Nächster Schritt: Analysiere den modularen Multiplikationsalgorithmus

Montgomerys Multiplikationsalgorithmus (I)

- weit verbreiteter, sehr effizienter Algorithmus zur Berechnung modularer Multiplikationen

- Laufzeit für Montgomery-Multiplikation $\in \{c, c+c_{ER}\}$

Zeit für Extrareduktion
(= zusätzliche Subtraktion)

Montgomerys Multiplikationsalgorithmus (II)

x Wortbreite der genutzten Arithmetik

$r = 2^x$, $R := r^t = 2^{xt} > n$

R Montgomery-Konstante

$n' := -n^{-1} \pmod r$

<u>Input:</u> $a = (a_{t-1}, \ldots, a_0)_r$, $b = (b_{t-1}, \ldots, b_0)_r < m$

<u>Output:</u> $abR^{-1} \pmod n$

Montgomerys Multiplikationsalgorithmus (III)

$s := 0$

for i=0 to t-1 do {

$u_i := (s_0 + a_i b_0)\, n'$ (mod r)

$s := (s + a_i b + u_i n) / r$

}

if $(s \geq n)$ then $s := s - n$ **ExtraReduktion (ER)**

return s (= abR^{-1} (mod n))

Montgomerys Multiplikationsalgorithmus (IV)

Unabhängig von der Wortbreite x gilt

Extrareduktion für Input a,b $\Leftrightarrow$

$$\frac{a}{n}\,\frac{b}{n}\,\frac{n}{R} + \frac{abn^* \pmod R}{R} \geq 1$$

<u>Fragestellung:</u> Eigenschaften bei zufälligem Input ?

Stochastisches Modell (II)

Montgomery-Konstante

$\Psi: Z \rightarrow \{0,1,\ldots,n-1\}, \qquad \Psi(x) := xR \pmod n$

$$S_{i,j+1} := \begin{cases} S_{i,j} * \Psi(y_i) / R + V_{i,j+1} \pmod 1 & \text{Mult. m. Basis} \\ S_{i,j}^2 * n / R + V_{i,j+1} \pmod 1 & \text{Quadrierung} \end{cases}$$

mit $V_{i,1}, V_{i,2}, \ldots$ iid gleichverteilt auf $[0,1)$

$\rightarrow S_{i,1}, S_{i,2}, \ldots$ iid gleichverteilt auf $[0,1)$

Stochastisches Modell (III)

Interpretiere Laufzeiten für modulare Operationen als Realisierungen von Zufallsvariablen $W_{i,1}, W_{i,2}, \ldots$

$W_{i,j+1} = c + c_{ER}$ (d.h. Extrareduktion) $\Leftrightarrow$

$$\begin{cases} S_{i,j} * \Psi(y_i) / R + V_{i,j+1} \geq 1 & \text{Mult. m. Basis} \\ S_{i,j}^2 * n / R + V_{i,j+1} \geq 1 & \text{Quadrierung} \end{cases}$$

Analysiere den stochastischen Prozess $W_{i,1}, W_{i,2}, \ldots$

Fundamentale Eigenschaften

$$E(W_{i,j}) := c + c_{ER} = \begin{cases} \frac{1}{2}\,\frac{n}{R}\,\frac{\Psi(y_i)}{n} & \text{Multiplikation mit Basis} \\ \frac{1}{3}\,\frac{n}{R} & \text{Quadrierung} \end{cases}$$

Prob. für ER

- Die Zufallsvariablen $W_{i,1}, W_{i,2}, \ldots$ sind weder unabhängig noch stationär.
- Nachfolger $W_{i,j}$ und $W_{i,j+1}$ sind negativ korreliert.
- Es gilt eine Version des zentralen Grenzwertsatzes für abhängige Zufallsvariablen.

Effizienz

Angriffsziel: Cascade Chip (ARM7M Risc Prozessor, s&m, Montgomery-Algorithmus, **ungeschützt gegen Laufzeitangriffe**)

Koeune, Quisquater et al. (Cardis 1998):
200.000 – 300.000 Zeitmessungen (512-Bit Schlüssel)

Schindler, Koeune, Quisquater (Crypto & Coding 2001)
Schindler (2002):
identische Hardware, neue Schätzer

5000 Zeitmessungen

Erfolgsrate:	74 %	(512-Bit Schlüssel)
Erfolgsrate (Simulation):	85 %	(512-Bit Schlüssel)

Schutzmaßnahmen

- konstante Laufzeiten für alle mod. Multiplikationen
- Blinding-Techniken (Randomisierung)
- Chinesischen Restsatz (CRT) nutzen (???)

Der CRT galt mehrere Jahre als immun gegen Laufzeitangriffe.

Ohne weitere Gegenmaßnahmen verhindert der CRT zwar diese Laufzeitattacke, ermöglicht aber einen noch weitaus effizienteren Laufzeitangriff!

Beispiel 2: Laufzeitangriffe gegen RSA mit CRT

Angriff gegen CRT, Montgomery-Algorithmus, ohne Blinding:

- Schindler (2000)
  - „Faktorisieren durch Laufzeitunterschiede“
  - 1024 Bit-Modul (s&m): 300 Laufzeitmessungen

Ohne die stochastische Modellierung des Exponentiationsalgorithmus und die Analyse des stochastischen Prozesses $W_{i,1}$, $W_{i,2}$, ... wäre dieser Angriff nicht entdeckt worden.

Beispiel 3: Laufzeitangriffe gegen OpenSSL

- Brumley, Boneh (2003): Adaption des Angriffs aus Beispiel 2 ermöglicht Remote-Angriff gegen OpenSSL-Server über lokales Netzwerk

- Folge: OpenSSL implementiert Basisblinding als Default-Gegenmaßnahme. Basisblinding verhindert *reine* Laufzeitattacken.

- Aciicmez, Schindler, Koc (2005): Effizienzsteigerung der Attacke um Faktor > 10 (ohne Blinding)

Beispiel 4: Kombinierte Laufzeit- und Powerattacken

Walter, Thomson (2001), Schindler (2002), Schindler, Walter (2003)

<u>Annahmen:</u>

- Chipkarte berechnet y^d (mod n)
  MM-Alg. + tabellengestützter Exponentiations-algorithmus (speichert kleine Potenzen der Basis)
- Gegenmaßnahmen (u.a. Basisblinding) verhindern *reine* Laufzeitangriffe und *reine* Powerangriffe.
- Strommessungen: → Anfang und Ende der einzelnen MM-Multiplikationen (→ Extrareduktionen)

Effizienz

Walter, Thomson (2001): nur 2-Bit-Tabellen

Schindler (2002): b-Bit-Tabellen ($b \geq 2$)

b=2: Effizienzsteigerung um Faktor >5

<u>Gegenmaßnahmen:</u>

- Exponentenblinding
- MM-Variante mit konstanten Laufzeiten
  (Nachteil: benötigt zusätzliche Hardware-Multiplier)

Beispiel 5: Cache-Attacken gegen Blockchiffren

- Typisches Angriffsziel: Blockchiffre (z.B. AES) auf PC / Server

- Grundidee:
  - teilschlüsselabhängige Tabellen werden in Cache geladen
  - → Schlüssel

OpenSSL (Stand: 1. Hj. 2007)

- RSA: CRT mit Montgomery-Algorithmus
- Modulare Exponentiation:
  - Fixed Windows (5-Bit-Fenster für 1024-Bit RSA)
  - Sliding Windows (5-Bit-Fenster für 1024-Bit RSA)
- Gegenmaßnahmen
  - Laufzeitangriffe: Basisblinding (seit 2003)
  - Cache und Branchprediction-Angriffe: ...

Allgemein: Fixed-Windows-Exp.-Algorithmus

Berechne y^d (mod n)

Tabelle speichert Potenzen: $y, y^2 (\text{mod } n), \ldots, y^{31} (\text{mod } n)$

<u>Beispiel:</u> d = 10010011000010110011...

d = 10010 01100 00101 10011 ...

„Operation“: $QQQQQM_{18}$ $QQQQQM_{12}$ $QQQQQM_5$ $QQQQQM_{19}$ ...

Folge von Operationen ⟷ d

Q = Quadrierung

M_j = Multiplikation mit j-tem Tabelleneintrag

Beispiel 6: Instruktions-Cache-Attacke gegen OpenSSL

- Aciicmez, Schindler (2007)
- Target: damals aktuelle OpenSSL-Version 0.9.8.e
- Status: OpenSSL ist inzwischen gepatcht.

Angriff: Grundidee

- Phase 1: Instruktions-Cache-Attacke
  - mod. Exponentiationsroutine und ein Spionage-prozess greifen auf (denselben) Instruktions-Cache zu
  - Spionageprozess liefert (fehlerbehaftete) Information, welche modulare Operationen ERs benötigen.

Beispiel: (1 = ER / 0 = keine ER)

Basis-Nr	Tabelle: Initialisierung 1	2	31	Exponentiationsphase 1	2	i	.....
1	1	0	1	0		0	1 .....
2	0	1.....	1	0		0	1 .....
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
N-1	0	0	0	1		0	0 .....
N	0	0	1	0		1	1 .....

„Quelle“ des Angriffs

Operation i ?

Angriff: Grundidee (II)

- Phase 1: Instruktions-Cache-Attacke
  - mod. Exponentiationsroutine und ein Spionageprozess greifen auf (denselben) Instruktions-Cache zu
  - Spionageprozess liefert (fehlerbehaftete) Information, welche modularen Operationen ERs benötigen.
- Phase 2:
  - a) Bestimme optimale Entscheidungsstrategie
  - b) Rate Operationstypen (→ Kandidat für d)
  - c) Verifikaton bzw. Fehlerkorrektur der Schätzer

Phase 2

- Analysiere den stochastischen Prozess $W'_{i,1}$, ..., $W'_{i,31}$, $W_{i,1}$, $W_{i,2}$, ...

(vgl. Beispiel 1) Stochastischer Prozess beschreibt Tabellenaufbau + Exponentiationsphase

→ optimale Entscheidungsstrategie

- Die optimale Entscheidungsstrategie kann 4-5 % fehlerhafte ER-Klassifikationen aus Angriffsphase 1 kompensieren.

Empirische Resultate

- 1024-bit RSA, 5-Bit-Fixed-Windows Exponentiation
- Phase 1: “Pool” von ER-Vektoren für N = 2000 zufällig gewählte Basen; jede Messung wurde 16 Mal wiederholt
- Die optimale Entscheidungsstrategie wurde auf zufällig gewählte Teilmengen von N_1 ER-Vektoren angewandt

N_1	guessing errors (average)	# attacks with ≤ 1 guessing errors
500	1.90	44/100
600	0.83	85/100
700	0.25	99/100
800	0.12	99/100
900	0.01	100/100
1000	0.02	100/100

- Stichprobenumfang N_1: erfordert Berechnung von $32*N_1$ vielen 34-dimensionalen Integralen

Analogie

- kombinierte Laufzeit-/Powerattacke (2002)
  - Powerattacke: Wo finden ERs statt?
  - Bestimme optimale Entscheidungsstrategie
- Angriff gegen OpenSSL (2007)
  - Instruktions-Cache-Attacke: Wo finden ERs statt?
  - Bestimme optimale Entscheidungsstrategie (starke Analogie zu kombinierter Laufzeit- / Powerattacke von 2002)

Chronologie (I)

- 1996, 1998:
  Laufzeitattacken gegen RSA ohne CRT
  Schutz: Basisblinding, Exponentenblinding
- 2000 - 2002:
  Optimierung dieser Angriffe
  Laufzeitattacke gegen RSA mit CRT
- 2002: kombinierte Laufzeit- und Powerattacke:
  Basisblinding verhindert reine Laufzeitangriffe,
  aber nicht lokale (kombinierte) Laufzeitangriffe
  Schutz: Exponentenblinding,
  konstante Laufzeiten für Montgomery-Multiplikation

Chronologie (II)

In der Welt der Softwareimplementierungen ...

- bis 2003: keine Default-Gegenmaßnahmen gegen Laufzeitangriffe bei OpenSSL
- 2003: Remote-Laufzeitangriff gegen OpenSSL
  Maßnahme: OpenSSL: Basisblinding als Default-Einstellung
- 2007: Instruction-Cache-Attacke gegen OpenSSL
  Maßnahme: OpenSSL: konstante Laufzeit für Montgomery-Multiplikationen

Beobachtungen (I)

- Die hardwarenahe Krypto-Community verliert an Angriffen, die (vermeintlich) durch bestimmte Gegenmaßnahmen abgewehrt werden können, oftmals schnell das Interesse.
- Erkenntnisse über Seitenkanalangriffe gegen Hardware werden von Softwareentwicklern kaum wahrgenommen.
- Tatsächlich kann man viele Ideen übertragen, wenngleich die konkrete Ausgestaltung der Seitenkanalangriffe sehr unterschiedlich sein kann.

Beobachtung (II)

Das hat in der Vergangenheit dazu geführt, dass Gegenmaßnahmen z.T. nur schrittweise verbessert wurden.
Etliche bekannte (publizierte) Angriffe unterstützen die Vermutung, dass dies insbesondere ein Problem von Softwareimplementierungen ist.

Fachliches Resümee (I)

- Die Resistenz gegen Seitenkanalangriffe ist für Chipkarten wie für Sicherheitssoftware von herausragender Bedeutung.
- In den nächsten Jahren ist vor allem bei Softwareimplementierungen mit neuen Seitenkanalangriffen zu rechnen.

Fachliches Resümee (II)

- Um wirksame und vertrauenswürdige Schutz-maßnahmen entwickeln zu können, sollten Designer das wahre Risikopotential von Seiten-kanalangriffen kennen.

Stochastische und statistische Ansätze haben sich hierfür als äußerst wirksam erwiesen.

Kontakt

Bundesamt für Sicherheit in der
Informationstechnik (BSI)

Prof. Dr. Werner Schindler
Godesberger Allee 185-189
53175 Bonn

Tel: +49 (0)1888-9582-5652
Fax: +49 (0)1888-10-9582-5652

Werner.Schindler@bsi.bund.de
www.bsi.bund.de
www.bsi-fuer-buerger.de