XTR Public Key System

Sebastian Schleemilch

***Zusammenfassung*—Viele der heutigen Kryptosystemen basieren auf der Unlösbarkeit des diskreten Logarithmus Problems. Zu dieser Kategorie gehört DSA für die Erzeugung von digitalen Signaturen, der Diffie-Hellman Schlüsseltausch sowie das ElGamal-Signaturverfahren. Das hier vorgestellte XTR-Verfahren kann auf alle gängigen Verfahren angewendet werden die auf dem diskreten Logarithmus Problem über einer multiplikativen Gruppe eines endlichen Körpers basieren. Das Ziel von XTR ist eine kompaktere Darstellung des endlichen Körpers mit Hilfe einer Untergruppe. Um die Alltagstauglichkeit von Kryptosystemen zu erhöhen, sollten sie so unauffällig wie möglich in die Anwendung integriert werden können. XTR leistet hier einen guten Beitrag. Mit XTR sind die zu übertragenen Schlüssellängen (=Datenmenge), bei nachweislich gleichbleibender Sicherheit, kürzer. Außerdem sind die erforderlichen Berechnungen für die Kryptosysteme effizienter als bei vergleichbaren Verfahren. Somit bietet sich das Verfahren auch für die Verwendung in Low-Power-Anwendungen wie z.B. SmartCards an.**

I. Einleitung

Der Wunsch nach der Verschlüsselung von digitalen Daten wird sowohl im privaten als auch im geschäftlichen Bereich immer größer. Ausschlaggebend für die Verwendung von Verschlüsselungstechnologien sind deren Sicherheit sowie deren Benutzbarkeit. Grundsätzlich gibt es zwei Verfahren um eine Ende-zu-Ende Verschlüsselung umzusetzen. Es gibt die symmetrische Verschlüsselung, bei der sich Sender *A* und Empfänger *B* auf einen gemeinsamen privaten Schlüssel einigen. Diesen Schlüssel verwenden die Teilnehmer um geheime Nachrichten auszutauschen. Das Problem liegt hier bei dem sicheren Austausch dieses Schlüssels, ohne dass ein Angreifer die Möglichkeit hat den Schlüssel mitzulesen. Der orthogonale Ansatz ist die Verwendung von Schlüsselpaaren pro Gesprächsteilnehmer. Sowohl Teilnehmer *A* als auch *B* erzeugen jeweils einen öffentlichen sowie einen privaten Schlüssel. Der öffentliche Teil des Schlüsselpaares kann in einer vertrauenswürdigen, für jeden Teilnehmer zugänglichen, Datenbank gespeichert werden um „Man-in-the-middle"'-Angriffe zu vermeiden. Möchte *A* eine Nachricht an *B* schicken, besorgt sich *A* den öffentlichen Schlüssel von *B* und verschlüsselt mit diesem die Nachricht *m*. Nur *B* kann diese Nachricht mit dem nur *B* bekannten privaten Schlüssel lesen.

II. Mathematische Grundlagen der Gruppen und der Diffie-Hellman Schlüsselaustausch

Diffie-Hellman ist ein Schlüsselaustauschverfahren (DH) mit dessen Hilfe Teilnehmer *A* und *B* ihren geheimen Schlüssel für eine zukünftig symmetrische Verschlüsselung über einen unsicheren Übertragungskanal austauschen können. Obwohl ein Angreifer möglicherweise die gesamte Kommunikation mithört, ist er nicht in der Lage sich den geheimen Schlüssel zu erzeugen. Dieses Verfahren und die dabei verwendeten Elemente sind eine gute Grundlage für das Verständnis von XTR, da sich XTR auch für dieses Verfahren eignet.

Für den DH-Schlüsselaustausch benötigt man eine Primzahl p, sowie eine *multiplikative zyklische Gruppe* Z_p^* über dieser Primzahl.

Eine Gruppe im mathematischen Sinn besteht allgemein aus Elementen die sich Verknüpfen lassen, wie z.B. durch die Addition „+", Multiplikation „·" oder der Division „%" aber auch durch komplexere Verknüpfungen wie der Berechnung „$(a+b) \bmod c$". Die Gruppe muss außerdem folgende Eigenschaften besitzen [3]:

- Assoziativität bezüglich der Verknüpfung $\circ$: $a \circ (b \circ c) = (a \circ b) \circ c$
- Es muss für jedes Element ein neutrales Element *b* geben so dass mit der Verknüpfung $\circ$ gilt: $a \circ b = a$
- Es muss für jedes Element ein inverses Element *b* geben so dass mit der Verknüpfung $\circ$ gilt: $a \circ b =$ *neutrales Element*

So besteht beispielsweise Z_{10} aus den Elementen $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Diese Elemente bilden bezüglich der Addition mod 10 eine Gruppe, nicht aber bezüglich der Multiplikation mod 10. Hier gibt es für einige Elemente kein inverses Element um das neutrale Element der Multiplikation, also die Zahl 1 zu errechnen. Sei $x \in \{0, 2, 4, 5, 6, 8\}$ und $y \in Z_{10}$. Für eine multiplikative Gruppe muss für alle Elemente die Eigenschaft $x * y = 1 \bmod 10$ gelten. x enthält in diesem Beispiel genau die Elemente ohne Lösung. Entfernt man die Elemente x aus Z_{10}, verbleibt dann also *multiplikative Gruppe* $Z_{10}^* = \{1, 3, 7, 9\}$ [4].

Allgemein besteht eine Gruppe Z_n^* aus Elementen die zu *n* Teilerfremd sind. Ist *n* eine Primzahl *p*, so besitzt diese Gruppe also $n - 1$ Elemente (z.B. $Z_7^* = \{1, 2, 3, 4, 5, 6\}$).

Eine zyklische Gruppe liegt vor, wenn die Gruppe aus den Potenzen eines einzelnen Elementes erzeugt werden kann. Beispielsweise lässt sich die zyklische multiplikative Gruppe $Z_7^* = \{1, 2, 3, 4, 5, 6\}$ durch Exponenten des Elements $5 \bmod 7$ erzeugen ($\{5^1, 5^2, 5^3, 5^4, 5^5, 5^6\} = \{5, 4, 6, 2, 3, 1\} = \{1, 2, 3, 4, 5, 6\}$). Die Zahl *5* aus diesem Beispiel wird Generator *g* genannt. Eine Gruppe Z_n^* ist außerdem auf jeden Fall zyklisch, falls $n = 2, 4, p^k, 2p^k$. Für XTR ist außerdem der Begriff einer Untergruppe wichtig. Eine Untergruppe besteht aus einer Teilmenge der Obergruppe. Sie muss jedoch in sich bezüglich der gewählten Verknüpfung wieder eine eigene Gruppe bilden.

Für das DH-Schlüsselaustauschverfahren einigen sich die Kommunikationspartner *A* und *B* nun über einen unsicheren abhörbaren Kommunikationskanal auf einen Generator *g* und eine Primzahl *p* und können so die zuvor eingeführte zyklisch multiplikative Gruppe Z_p^* bilden. Teilnehmer *A* berechnet nun $X = g^a \bmod p$ mit einem zufälligen Wert für *a* und schickt *X* über den unsicheren Kanal zu *B*. Teilnehmer *B* berechnet

$Y = g^b \bmod p$ und schickt *Y* zu *A*. Teilnehmer *A* und *B* können sich jetzt den geheimen Schlüssel $Y^a = g^{ab} \bmod p$ bzw. $X^b = g^{ab} \bmod p$ berechnen, der zukünftig für eine symmetrische Verschlüsselung verwendet werden kann.

Ein möglicher Angreifer konnte während des Vorgangs also folgendes über den unsicheren Kanal mithören: *g*, *p*, $X = g^a \bmod p$, $Y = g^b \bmod p$. Um den geheimen Schlüssel $g^{ab} \bmod p$ zu rekonstruieren, müsste der Angreifer also entweder a aus dem Wert $g^a \bmod p$ oder b aus dem Wert $g^b \bmod p$ berechnen.

Diese Ermittlung des Wertes ist als das *diskrete Logarithmus Problem* bekannt. Die Sicherheit des DH-Schlüsselaustausches beruht also auf der Schwierigkeit $g^x \bmod p$ nach *x* aufzulösen. Für die Bestimmung von x ist hier noch kein effizientes Verfahren bekannt. Das Problem wird verschärft je größer die Primzahl p gewählt wird.

Allein für den Schlüsselaustausch werden also einige Daten über einen Kanal geschickt. Nach heutigem Stand gilt dieses Verfahren ab einer Mächtigkeit von 1024 Bit für Z_p^* als sicher.

Der DH-Schlüsselaustausch dient als Beispiel für eine mögliche Anwendung von XTR. Hier wird eine, für eine ausreichende Sicherheit, große zyklische multiplikative Gruppe verwendet um das diskrete Logarithmusproblem als unlösbar zu gestalten. XTR verallgemeinert das Konzept des zyklisch multiplikativen Gruppe mit der Einführung von Körpern und beschreibt ein Verfahren, mit dem ein großer Körper auf einen kleineren abgebildet werden kann.

III. XTR zur Reduktion des Kommunikationsoverheads

A. Grundlagen für XTR - Galois-Körper GF

Das XTR Verfahren, „Efficient and Compact Subgroup Trace Representation“ beschäftigt sich mit der Reduktion des Kommunikationsoverheads bei möglichst gleichbleibender Sicherheit, also der Minimierung der zu übertragenden Bits bei angewandten Kryptografieverfahren wie z.B. dem kurz vorgestelltem DH-Schlüsselaustausches. Dieses Verfahren arbeitet mit endlichen Körpern, sogenannten Galois Körpern *GF*. Ein Körper ist strenger definiert als die zuvor eingeführte Gruppe. Ein Körper *K* besitzt die Addition „+“ und die Multiplikation „$\cdot$“ als Verknüpfungen mit den folgenden Eigenschaften:

- $(K, +)$ ist eine abelsche Gruppe mit neutralem Element 0, also wie die zuvor definierte Gruppe mit zusätzlichem Kommutativgesetz
- $(K\backslash\{0\}, \cdot)$ ist eine ablesche Gruppe mit neutralem Element 1
- Distributivgesetz für alle Elemente aus K

Ein endlicher Körper (Galois Körper, GF), ist ein Körper mit einer endlichen Anzahl von Elementen. Für jede Primzahl p und jede natürliche Zahl n existiert ein Körper mit p^n Elementen. Er wird mit F_{p^n} oder $GF(p^n)$ bezeichnet. Hierbei ist die Addition und Multiplikation wie gewohnt modulo p auszuführen (Ansonsten wäre der resultierende Körper nicht endlich, da eine Addition von zwei ausgewählten Elementen ein „neues“ Element des Körpers ergeben kann).

Wie bereits erwähnt, hängt die Sicherheit der Anwendung eines Verfahrens mit diskretem Logarithmus-Problem maßgeblich von der Anzahl der Elemente in so einem Körper $GF(p^n)$, also von p^n ab. Der XTR-Algorithmus ist in der Lage beliebige Exponenten des Generators g des Körpers $GF(p^6)$ über $GF(p^2)$ darstellen zu können. Außerdem kann die Exponentiation in diesem Teilkörper effizient berechnet werden.

B. Funktionsweise des XTR-Verfahrens

Die Artefakte von XTR sind wie erwähnt $GF(p^6)$ und $GF(p^2)$. Es wird jetzt ein Generator $g \in GF(p^6)$ mit der Ordnung $q > 6$ gewählt. Die Ordnung von g ist definiert als die kleinste ganze Zahl n für die gilt: $g^n \bmod p = 1$. Außerdem muss g den Wert $p^2 - p + 1$ teilen. Das so gewählte g erzeugt nun eine Untergruppe von $GF(p^6)$ mit der Ordnung q. Für XTR muss im Gegensatz zu vielen anderen Verfahren weder diese Untergruppe $< g >$, noch Elemente aus $GF(p^6)$ erzeugt werden. Stattdessen reicht eine Repräsentation von $GF(p^2)$ aus.

Die auszuwählende Primzahl p hat die Eigenschaft $p \equiv 2 \bmod 3$, $p \equiv 2 \bmod 3$ generiert also $GF(3)*$. Daraus folgt, dass sich Elemente aus $GF(p^2)$ mit $x_1\alpha + x_2\alpha^p = x_1\alpha + x_2\alpha^2$ darstellen lassen, wobei $x_1, x_2 \in GF(p)$ und α, α^p die Nullstellen des nicht reduzierbaren Polynoms $X^2 + X + 1$ sind.

Für Kryptosysteme müssen je nach Verfahren die verschiedensten Berechnungen durchgeführt werden. Für Elemente aus $GF(p^2)$ ergeben sich nun folgende Komplexitäten für $x, y \in GF(p^2)$:

- $x^p = x_1^p\alpha^p + x_2^p\alpha^{2p} = x_1\alpha + x_2\alpha^2$. Die Berechnung ist also kostenlos.
- x^2 kann auf zwei Quadrierungen und eine Multiplikation in $GF(p)$ zurückgeführt werden
- $x \cdot y$ benötigt drei Multiplikationen in $GF(p)$

Diese Analyse kann man ebenfalls für die Elemente $x, y \in GF(p^6)$ mit gleichbleibendem p durchführen. Hier zeigt sich ein deutlicher Berechnungskomplexitätsunterschied. So benötigt die Quadrierung in $GF(p^6)$ ca. 14 Multiplikationen in $GF(p)$, die Multiplikation $x \cdot y$ sogar 18.

Die Berechnungen in $GF(p^2)$ sind also wesentlich effizienter als in $GF(p^6)$.

$GF(p^2)$ (im weiteren als K bezeichnet) ist offensichtlich ein Unterkörper von $GF(p^6)$ (im weiteren Verlauf als L bezeichnet), da $K \subseteq L$. Das Paar L und K wird als Körpererweiterung bezeichnet $(L\backslash K)$.

Um die Elemente aus L ($GF(p^6)$) mit Elementen aus K ($GF(p^2)$) darzustellen, verwendet XTR die Spur(engl. Trace, Abkürzung: Tr) von L nach K. Mathematisch ist die Spur einer Körpererweiterung eine lineare Abblidung von L nach K, also die Zuordnung von Elementen des größeren Körpers L auf den kleineren Körper K.

In [1] wird gezeigt, dass alle Elemente von $g^n \in GF(p^6)$ mit der Hilfe der Spur $Tr(g^n)$ dargestellt werden können, wobei diese Spur in $GF(p^2)$ liegt. Es wird also nur noch mit den Spuren gerechnet. Um diese Darstellung in kryptographischen Verfahren verwenden zu können, muss $Tr(g^n)$ über ein bekanntes $Tr(g)$ bestimmt werden können (Entspricht der Exponentiation von einem bekannten g zu g^n). Es existiert ein Algorithmus, der folgende Berechnung für beliebige n

ermöglicht:
$S_n(Tr(g)) = (Tr(g^{n-1}), Tr(g^n), Tr(g^{n+1}))$
Hiermit lässt sich aus $Tr(g)$ u.a. ein beliebiges $Tr(g^n)$ bestimmen. Eine entsprechende Analyse zeigt außerdem, dass die Berechnung von $Tr(g^n)$ drei mal so schnell ist wie die Berechnung von g^n. Das Ergebnis eines beliebigen $Tr(g^n)$ befindet sich nachweislich im Körper $GF(p^2)$, die Elemente der Berechnung g^n aber in $GF(p^6)$. Das Sicherheitsniveau aufgrund der Mächtigkeit von $GF(p^6)$ bleibt also mit dieser Methode erhalten.

Das Ziel, eine größere Gruppe g^n mit einer kleineren Gruppe darzustellen wird also durch die Verwendung der Spur erreicht. In einigen kryptografischen Verfahren wird allerdings nicht nur eine Berechnung von g^n benötigt sondern meistens auch ein Produkt von zwei oder mehreren Exponentiationen von g, also beispielsweise $g^a \cdot g^b$. Bei der Verwendung der klassichen Repräsentation ist diese Rechnung trivial und einfach durchzuführen ($g^a \cdot g^b = g^{a+b}$). Die Berechnung über die Spur von g ist dabei schwieriger. Die Publikation [1] zeigt jedoch ein Verfahren mit dem $Tr(g^a \cdot g^b)$ effizient berechnet werden kann, sogar mit einem Effizienzzuwachs von 75% gegenüber der Berechnung von $g^a \cdot g^b$.

C. Wahl der benötigten Parameter für eine praktische Realisierung von XTR

Eine wichtige Frage bei der praktischen Umsetzung des XTR-Verfahrens betrifft die Wahl von p, der Untergruppengröße q von $GF(p^6)$ sowie einem geeigneten $Tr(g)$. Das Resultat wird als „XTR group“ bezeichnet. Nach [1] muss q das Polynom $p^2 - p + 1$ teilen. Verglichen wird hier die Wahl der Parameter mit einer zu RSA vergleichbaren Sicherheit von 1024 Bit. Aus p wird der Körper $GF(p^6)$ erzeugt. Dieser soll dementsprechend eine Mächtigkeit von ebenfalls 1024 Bit aufweisen. Somit ergibt sich für p eine ungefähre Länge von $1024/6 \approx 170$ Bit. Aufgrund aktueller kryptoanalytischen Methoden sollte p nicht viel kleiner als q gewählt werden, also beide ca. 170Bit. Die Parameter können über folgende Gleichungen ermittelt werden:

- Finde ein r mit den Eigenschaften: q ist eine Primzahl der Länge 170 Bit und $q = r^2 - r + 1$
- Danach muss ein $k \neq 1$ ermittelt werden für das gilt: $p = r + k \cdot q$, wobei p wieder eine Länge von ca. 170 Bit besitzen sollte und außerdem $p \equiv 2 \bmod 3$ gilt. Bei der Wahl von $k \equiv 1$ vereinfacht sich die Lösung des diskreten Logarithmus-Problems und ist somit zu vermeiden.
- Ermittlung eines $Tr(g)$ wie in [1] beschrieben.

Das „public key“ Element des XTR Verfahrens ist somit das Tripel $(p, q, Tr(g))$ und kann zwischen den Kommunikationspartnern ausgetauscht werden. Wie beschrieben besteht es aus dem finiten Feld (generiert aus p), der Ordnung der Untergruppe (q) sowie dem Generator $Tr(g)$. Das XTR Verfahren unterstützt auch Protokolle, die private Schlüssel vorsehen, sowie die digitale Signierung. Ein privater Schlüssel hat dann die Form $Tr(g^k)$, wobei k hier der geheime Teil ist (denn $Tr(g)$ ist öffentlich).

Die explizite Anwendung des XTR Verfahrens auf speizelle Kryptosysteme wie den DH-Schlüsselaustausch, die ElGamal Verschlüsselung sowie die Nyberg-Rueppel Signatur lässt sich in [1] nachlesen.

IV. Sicherheit von XTR

A. Diskreter Logarithmus der XTR-Artefakte

Das in II erwähnte Logarithmus Problem muss hinsichtlich XTR auf den diskreten Logarithmus in $GF(p^t)$ angepasst untersucht werden. Dabei sei $< \gamma >$ eine Untergruppe der multiplikativen Gruppe $GF(p^t)*$ der Ordnung ω. Es lassen sich verschiedene Probleme bezüglich des Diffie-Hellman Schlüsselaustausches definieren:

- Diffie-Hellman Problem (DH, auch CDH genannt): Die Berechnung von γ^{xy} aus γ^x und γ^y, geschrieben als $DH(\gamma^x, \gamma^y) = \gamma^{xy}$
- Diffie-Hellman Decision (DHD, auch DDH genannt): Die Entscheidung ob für $a, b, c \in < \gamma >, c = DH(a, b)$ gilt, also die Zuordnung von c zu gegebenen a und b
- Diskretes Logarithmus Problem (DL): Die Berechnung von x bei einem gegebenen $a = \gamma^x$, geschrieben als $x = DL(a)$

Es ist hier die allgemeine Vermutung dass die Unlösbarkeit des DL-Problems die Unlösbarkeit von DH und DHD impliziert. Eine Analyse der möglichen Angriffsmethoden zeigt die Abhängigkeit des DL-Problems von der Mächtigkeit von $< \gamma >$ sowie der Größe von ω. Aufgrund der Verwendung der in III-C eingeführten XTR group ist das Problem nur für diese Untergruppe zu lösen. In [1] ist jedoch gezeigt dass bei einer geeigneten Wahl von p und q von ca. 170 Bits das DL Problem in der XTR group schwieriger zu lösen ist als die Faktorisierung in RSA mit 1020 Bits.

In XTR wird jedoch mit den Spuren gerechnet so dass die Probleme DL, DH und DHD wie folgt speziell auf XTR umgeschrieben werden können.

- XTR-DH Problem: Berechnung von $Tr(g^{xy})$ aus $Tr(g^x)$ und $Tr(g^y)$, geschrieben $XDH(g^x, g^y) = g^{xy}$
- XTR-DHD: Entscheidung ob $a, b, c \in Tr(< g >), c = XDH(a, b)$ gilt
- XTR-DL: Bestimmung von x aus $a = Tr(g^x)$, geschrieben als $x = XDL(a)$

Bei einer genaueren Untersuchung zeigt sich hier, dass die Algorithmen zur Lösung von DL, DH oder DHD auch zur Lösung von XDL, XDH und XDHD mit überschaubarem Aufwand umgeschrieben werden können. In der Praxis genügt also ein XTR-DH Algorithmus um ein DL Problem zu lösen. Bei der Wahl der vorgeschlagenen Parameter aus III-C ist die Lösung aber nachweislich mit der Sicherheit von RSA zu vergleichen.

B. Seitenkanalangriffe auf XTR

Seitenkanalangriffe versuchen über die Beobachtung der physikalisch auftretenden Effekte eines rechnenden Prozessors, Rückschlüsse auf die ausgeführten Anweisungen zu erhalten. Oft anzutreffen sind die Seitenkanalangriffe, die den Energieverbrauch des Prozessors aufzeichnen und auf diese Art ausgeführte Befehle zu extrahieren. Zwei bekannte Verfahren sind hier„Simple Power Analysis (SPA)“ und „Differential

Power Analysis (DPA)“. SPA nutzt nur Informationen aus einem einzelnen beobachteten Effekt, DPA verwendet mehrere Messquellen. Da bei der Verwendung von Smart-cards die Energieversorgung über ein externes Lesegerät drahtlos erfolgt, lässt sich hier beispielsweise die Energieverbrauchskurve leicht ermitteln. Ein Angreifer kann nun analysieren, welche Energiekurven zu welcher Operation gehören und beim Ablauf des Algorithmus eine Operations-Spur erzeugen, also einen rekonstruierten Kontrollfluss.

Für die Seitenkanalangriffsanalyse bezüglich XTR, werden die benötigten Operationen untersucht. Da XTR auf der Berechnung der Spur $Tr(g^n) \in GF(p^2)$ basiert, wird die Abkürzung $c_n = Tr(g^n)$ eingeführt, wobei $c_0 = 3$. Die Berechnung von $Tr(g^n)$ wird „Single Exponentiation (XTR-SE)“ genannt. Es lassen sich als Konsequenz folgende Rechenregeln aufstellen:

- $c_{-n} = c_{np} = c_n^p$
- $c_{2n} = c_n^2 - 2c_n^p$, im weiteren als $XTRDBL(c_n)$ bezeichnet
- $c_{n+2} = c_1 \cdot c_{n+1} - c_1^p \cdot c_n + c_{n-1}$, im weiteren als $XTR\_C_{n+2}(c_{n-1}, c_n, c_{n+1}, c_1)$ bezeichnet
- $c_{2n-1} = c_{n-1} \cdot c_n - c_1^p \cdot c_n^p + c_{n+1}^p$, im weiteren als $XTR\_C_{2n-1}(c_{n-1}, c_n, c_{n+1}, c_1)$ bezeichnet
- $c_{2n+1} = c_n \cdot c_{n+1} - c_1 \cdot c_n^p + c_{n-1}^p$, im weiteren als $XTR\_C_{2n+1}(c_{n-1}, c_n, c_{n+1}, c_1)$ bezeichnet

Zuletzt lassen sich hier noch die benötigten Multiplikationen der einzelnen Operationen in $GF(p)$ ermitteln. $XTRDBL(c_n)$ benötigt eine Multiplikation, XTR_C_{n+2}, XTR_C_{2n-1} sowie XTR_C_{2n+1} jeweils drei Multiplikationen. [2]

1) SPA Seitenkanalangriff: Für eine Berechnung von XTR-SE werden die Operationen ($XTRDBL(c_n)$, $XTRDBL(c_n)$, XTR_C_{2n-1}) oder ($XTRDBL(c_n)$, $XTRDBL(c_n)$, XTR_C_{2n+1}) je nach zu berechnendem Wert wiederholt ausgeführt. Da aber XTR_C_{2n-1} sowie XTR_C_{2n+1} die gleiche Anzahl an Multiplikationen in $GF(p)$ benötigen, sind die ausgeführten Instruktionen unabhängig von dem geheimen Schlüssel (dem Exponenten aus $Tr(g^x)$). XTR-SE ist also unempfindlich und somit sicher gegenüber SPA Seitenkanalangriffen.

2) Address-Bit DPA Seitenkanalangriff (ADPA): ADPA nutzt die mögliche Korrelation eines geheimen Wertes und die verwendeten Register bei der Abarbeitung des Wertes. So befindet sich in der Berechnung von XTR-SE aus [1] der folgende Teilausschnitt wobei m_i das i-te Bit des geheimen Wertes darstellt. $T[x]$ ist ein Feld für die Zwischenspeicherung von Werten und $C[x]$ Adressen.

- Falls m_i = 0: Aktualisiere T[1] mit dem Register C[0] und T[2] mit dem Register C[1]
- Falls m_i = 0: Aktualisiere T[1] mit dem Register C[1] und T[2] mit dem Register C[2]

Verknüpft der Angreifer nun die DPA Analyse mit den entsprechenden Registern C[0] und C[1], so kann er sich den geheimen Wert m ermitteln. XTR-SE ist also hinsichtlich dieses Angriffes nicht sicher. [2]

3) Verdoppelungsattacke: Die Verdoppelungsattacke basiert auf dem Ansatz dass ein Angreifer zwar nicht den Wert X einer Operation (z.B $2 \cdot X$) ermitteln kann, jedoch kann er prüfen, ob zwei Operationen das gleiche Ergebnis liefern (Beobachtetes Ergebnis X = Beobachteten Ergebnis Y). S_n steht für die Berechnung des Tripels (c_{n-1}, c_n, c_{n+2}) bei gegebenem c_1 (Zur Erinnerung: $c_n = Tr(g^n)$). Jetzt werden die Operationen verglichen bei der Berechnung von einem S_n bei gegebenem c_1 und einem $\tilde{S}_n$ wenn $\tilde{c}_1 = c_2$. Der private Wert m wird im XTR-SE Algorithmus mit einer For-Schleife mit Index i abgearbeitet. Hierbei lässt sich beobachten dass die XTRDBL Operation im Schritt i bei der Berechnung von S_n im Falle eines 0-Bits von m die gleiche Operation wie bei der Berechnung von $\tilde{S}_n$ im Schritt $i + 1$ ist. Mit der Hilfe dieses Verfahrens lässt sich als Konsequenz mit zwei Anfragen der private Teil m ermitteln. [2]

V. Fazit

Wie in dieser Arbeit zusammengefasst, ist XTR ein geeignetes Verfahren um effiziente Verschlüsselung durchgängig verwenden zu können. Bei einer immer weiter steigenden Rechenleistung von Desktop-Rechnern sowie Mobiltelefonen sollte man meinen, die benötigte Rechenleistung eines Verschlüsselungsverfahren spiele keine Rolle. In Hinblick auf Smard-Cards, NFC-Anwendungen und ähnliches ist die Effizienz sowie die erforderliche Bandbreite jedoch ein entscheidendes Kriterium. Der komplexe Sachverhalt des Hintergrundes von XTR kann für die praktische Anwendung aufgrund einfacher Algorithmen gut abstrahiert werden so dass sich die Einstiegshürde bei der Verwendung des Verfahrens reduziert. Auch die Wahl geeigneter Schlüssel wird dem Implementierendem weitestgehend abgenommen sodass die Fehleranfälligkeit für eine falsche Implementierung des Kryptografieverfahrens minimiert wird. Der hohe Effizienzzuwachs der Rechenoperationen in der XTR Darstellung von bis zu 600% beeindruckt vor allem mit dem Hintergrund der nachweislich äquivalenten Sicherheit gegenüber vergleichbaren Verfahren. Die Unempfindlichkeit von XTR gegenüber SPA Seitenkanalattacken sprechen außerdem für das Verfahren. Mit möglichen, hier nicht vorgestellten, Gegenmaßnahmen für einen ADPA sowie einen Verdoppelungsangriff, präsentiert sich XTR als ein solides, resourcenschonendes und vor allem sicheres Verschlüsselungsverfahren mit einem breiten Anwendungsgebiet.

Literatur

[1] A. Lenstra and E. R. Verheul *The XTR Public Key System*, Mendham, N.A and Eindhofen, Netherlands 2000.

[2] D. Hand, J. Lin and K. Sakurai *On Security of XTR Public Key Cryptosystems Against Side Channel Attacks*, Korea, Seoul and Japan, Fukuoka 2004

[3] H.W. Lang *Mathematische Grundlagen - Gruppe*, Available from: „http://www.iti.fh-flensburg.de/lang/algorithmen/grundlagen/gruppe.ht 28.08.2000

[4] H.W. Lang *Mathematische Grundlagen - multiplikative Gruppe modulo n*, Available from: „http://www.iti.fh-flensburg.de/lang/krypto/grund/gruppezn.htm“, 28.08.2000