

Breaking Ed25519 in WolfSSL

Luis Kress, Johannes Hausmann

Technische Hochschule Bingen

23-10-2019



Beispiel

Digitale Signatur und Verschlüsselung I

- ▶ Pendant der schriftlichen Signatur
- ▶ Dokument → Erklärung, Vereinbarung
- ▶ Nachweis
 - ▶ Inhalt des Dokument (Unterzeichner)
 - ▶ Verifikation (Empfänger)
- ▶ Signatur ausschließlich durch Unterzeichner
- ▶ Verifikation soll jedem möglich sein
siehe (Hühnlein and Korte 2006)

Digitale Signatur und Verschlüsselung II

1. Eine digitale Signatur ist ein String, welcher eine Nachricht mit einer Entität verbindet
2. Algorithmus zur Signaturerzeugung
3. Algorithmus zur Signaturverifikation
4. Signatur Schema (signature scheme) → Erzeugung & Verifikation
5. Signaturprozess → Formatierung der Daten in signierbare Nachrichten
6. Verifikationsprozess
siehe Katz et al. (1996)

Digitale Signatur und Verschlüsselung III

- ▶ Realisierung durch asymmetrische Kryptoalgorithmen
- ▶ Message
- ▶ K_{Priv}
- ▶ K_{Pub}
- ▶ Einwegfunktion

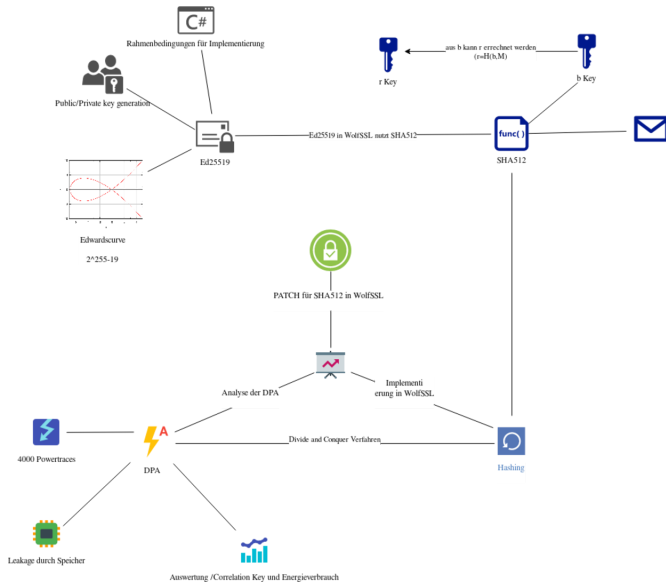
$$f(K_{Priv}) = K_{Pub}$$

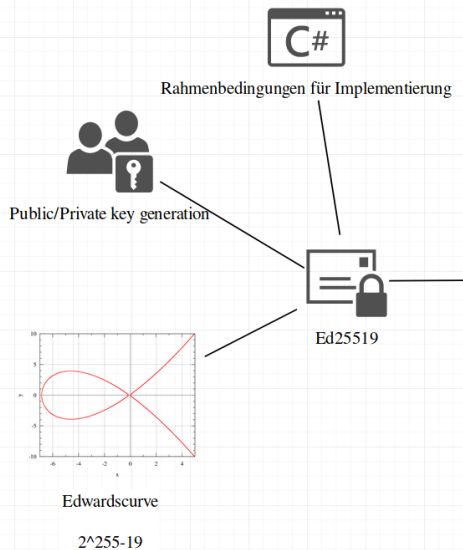
- ▶ inverse Funktionen
 - ▶ Signatur (Message, K_{Priv})
 - ▶ Verifikation (Message, Signatur, K_{Pub})
- ▶ K_{Pub} in öffentlichem Verzeichnis
siehe (Hühnlein and Korte 2006)

Beispiel für Verwendung von Digitale Signaturen

- ▶ SSL Zertifikat (CA)
- ▶ Software Installation auf Linux / BSD Systemen
- ▶ Elektronische Steuerklärung

```
RPM-GPG-KEY-fedora-21-s390x      RPM-GPG-KEY-fedora-iot-armhfp
RPM-GPG-KEY-fedora-21-secondary  RPM-GPG-KEY-fedora-iot-i386
RPM-GPG-KEY-fedora-21-x86_64     RPM-GPG-KEY-fedora-iot-ppc64le
RPM-GPG-KEY-fedora-22-aarch64    RPM-GPG-KEY-fedora-iot-s390x
RPM-GPG-KEY-fedora-22-armhfp     RPM-GPG-KEY-fedora-iot-x86_64
RPM-GPG-KEY-fedora-22-i386       RPM-GPG-KEY-fedora-modularity
RPM-GPG-KEY-fedora-22-ppc64      RPM-GPG-KEY-rpmfusion-free-fedora-30
RPM-GPG-KEY-fedora-22-ppc64le    RPM-GPG-KEY-rpmfusion-free-fedora-30-primary
RPM-GPG-KEY-fedora-22-primary    RPM-GPG-KEY-rpmfusion-free-fedora-31
RPM-GPG-KEY-fedora-22-s390      RPM-GPG-KEY-rpmfusion-free-fedora-31-primary
RPM-GPG-KEY-fedora-22-s390x      RPM-GPG-KEY-rpmfusion-free-fedora-32
RPM-GPG-KEY-fedora-22-secondary  RPM-GPG-KEY-rpmfusion-free-fedora-32-primary
RPM-GPG-KEY-fedora-22-x86_64     RPM-GPG-KEY-rpmfusion-free-fedora-latest
RPM-GPG-KEY-fedora-23-aarch64    RPM-GPG-KEY-rpmfusion-free-fedora-rawhide
RPM-GPG-KEY-fedora-23-armhfp     RPM-GPG-KEY-rpmfusion-nonfree-fedora-30
RPM-GPG-KEY-fedora-23-i386       RPM-GPG-KEY-rpmfusion-nonfree-fedora-30-primary
RPM-GPG-KEY-fedora-23-ppc64      RPM-GPG-KEY-rpmfusion-nonfree-fedora-31
RPM-GPG-KEY-fedora-23-ppc64le    RPM-GPG-KEY-rpmfusion-nonfree-fedora-31-primary
RPM-GPG-KEY-fedora-23-primary    RPM-GPG-KEY-rpmfusion-nonfree-fedora-32
RPM-GPG-KEY-fedora-23-s390      RPM-GPG-KEY-rpmfusion-nonfree-fedora-32-primary
RPM-GPG-KEY-fedora-23-s390x      RPM-GPG-KEY-rpmfusion-nonfree-fedora-latest
RPM-GPG-KEY-fedora-23-secondary  RPM-GPG-KEY-rpmfusion-nonfree-fedora-rawhide
RPM-GPG-KEY-fedora-23-x86_64     RPM-GPG-KEY-zfsonlinux
```





ECDSA & Ed25519

- ▶ Signaturverfahren basierend auf Elliptic Curve Cryptography (ECC)
 - ▶ Basis ist eine Punktruppe einer Elliptischen Kurve

ECDSA & Ed25519

- ▶ Signaturverfahren basierend auf Elliptic Curve Cryptography (ECC)
 - ▶ Basis ist eine Punktruppe einer Elliptischen Kurve
- ▶ ECDSA ist vorbereitet

ECDSA & Ed25519

- ▶ Signaturverfahren basierend auf Elliptic Curve Cryptography (ECC)
 - ▶ Basis ist eine Punktruppe einer Elliptischen Kurve
- ▶ ECDSA ist vorbereitet
 - ▶ EdDSA (Edwardscurve)

ECDSA & Ed25519

- ▶ Signaturverfahren basierend auf Elliptic Curve Cryptography (ECC)
 - ▶ Basis ist eine Punktgruppe einer Elliptischen Kurve
- ▶ ECDSA ist vorbereitet
 - ▶ EdDSA (Edwardscurve)
 - ▶ Ed25519 (Edwardscurve 25519)

ECDSA & Ed25519

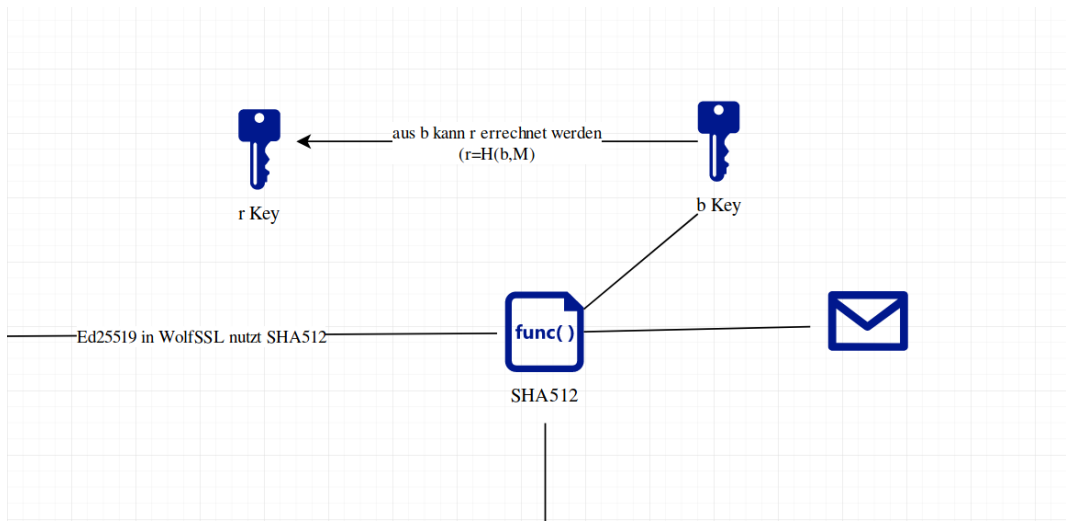
- ▶ Signaturverfahren basierend auf Elliptic Curve Cryptography (ECC)
 - ▶ Basis ist eine Punktgruppe einer Elliptischen Kurve
- ▶ ECDSA ist vorbereitet
 - ▶ EdDSA (Edwardscurve)
 - ▶ Ed25519 (Edwardscurve 25519)
- ▶ 160bit ECC Schlüssel = 1200bit RSA Schlüssel
 - ▶ Speicherverbrauch, Energieverbrauch (IoT)
siehe (Hühnlein and Korte 2006; Susella 2018)

ECDSA & Ed25519

- ▶ Verwendung (Susella 2018)
 - ▶ OpenSSH
 - ▶ WolfSSL / OpenSSL / LibreSSL / GnuTLS
 - ▶ Tor Protokoll
 - ▶ DNS Protokolle
 - ▶ Signal Messenger Protokoll



Tafelbild



Secure Hash “SHA512”

- ▶ Ed25519 nutzt SHA512
 - ▶ Merkle–Damgård Konstruktion
 - ▶ Erweiterung um Davies-Meyer
 - ▶ SHA-2 Familie (SHA256,SHA512) → Bitlänge des Hash
- ▶ Auxiliary Schlüssel b

Message M (Länge n) → SHA512 Funktion → 512 Bit Ausgabe

(Susella 2018)

Secure Hash “SHA512” II

Algorithm 3. Merkle Damgård

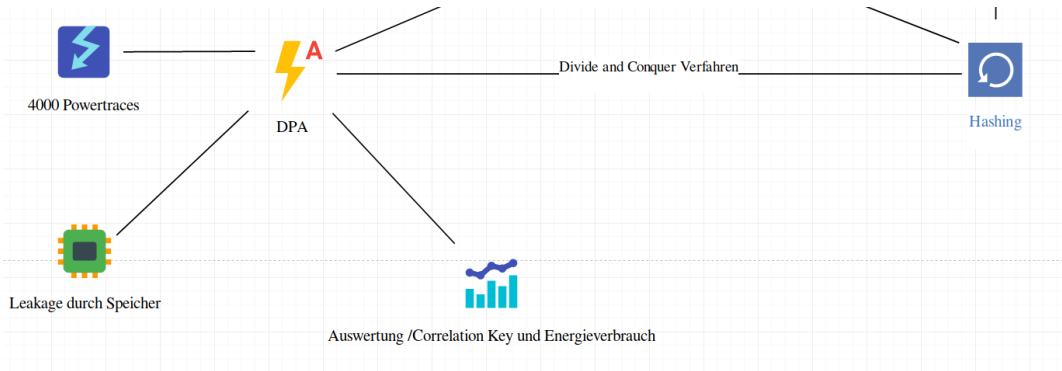
Input: Message M with $0 \leq \text{bit-length} < 2^{128}$

Output: Hash value of M

- 1: Pad message M by appending an encoding of the message length
 - 2: Initialize chaining value CV with constant IV
 - 3: Split padded message M into blocks
 - 4: **for all** blocks M_i **do**
 - 5: $CV_{i+1} \leftarrow CF(CV_i, M_i)$
 - 6: **end for**
 - 7: **return** $H \leftarrow CV$
-

Secure Hash “SHA512” III

- ▶ Ausgabe 512 Bit
- ▶ Größe des internen Status 512 Bit
- ▶ Blockgröße 1024 Bit
 - ▶ 16 Wörter
 - ▶ Wortgröße von 64 Bit
- ▶ 80 Runden
- ▶ Operationen auf Status
 - ▶ AND / XOR
 - ▶ Addition (mod 2^{64}) (Susella 2018)



Angriff auf Ed25519

- ▶ **Key Recovery** Attacke
 - ▶ Energieverbrauch eines SOC's
- ▶ Angriff bei Berechnung der "flüchtigen" Schlüssel
 - ▶ von Interesse ist Schlüssel b
- ▶ Hilfsschlüssel r bekannt
 - ▶ Scalar a , Hilfsschlüssel $b \rightarrow$ manipulierte Signaturen

Angriff auf Ed25519 II

- ▶ Differential Power Analysis (DPA)
 - ▶ SDA → Abhängigkeit Daten und Energieverbrauch
 - ▶ Energieverbrauch an einem Punkt der Encryption
- ▶ Zwischenwert (Intermediate Value)
 - ▶ Value mit bekanntem Teil/Message
 - ▶ Wert als Funktion darstellbar

$$S(M, k) = \textit{Value}$$

Angriff auf Ed25519 III

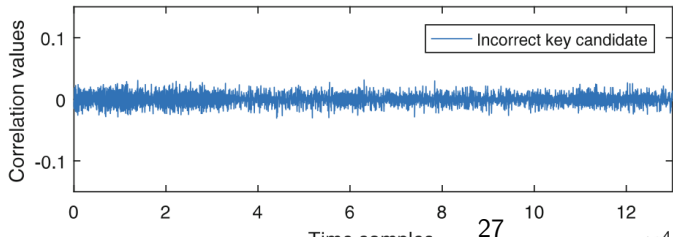
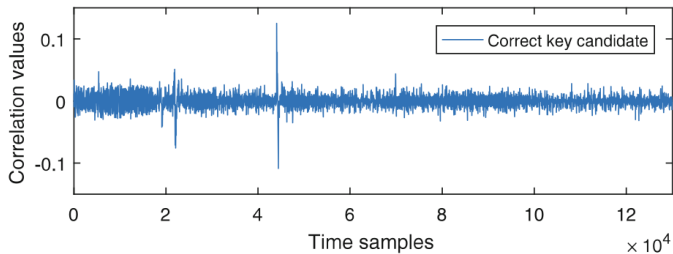
- ▶ 64 bit unbekannte Bits
 - ▶ 2^{64} mögliche Schlüssel
- ▶ Divide-and-Conquer Strategie
 - ▶ 8 Bit \rightarrow 256 mögliche Schlüssel

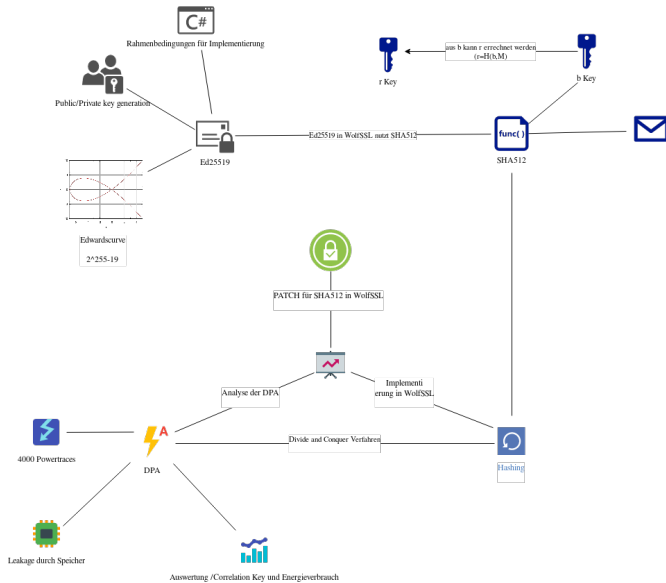
$$S(M, k^*)_{k_{16}, \text{ bit } 0-7} \leftarrow ((\sigma_1(w[14]) + w[9]) \bmod 2^8) + k^*$$

- ▶ Hamming Weight wird berechnet (Anzahl Traces X Key Kandidaten)
- ▶ Pearson Korrelation der Zeit (Time Samples, Hamming Weight)

Zu jedem Schlüssel Kandidaten muss jedes
Time sample eines Traces zugeordnet werden
 \rightarrow korrekte Ausrichtung für jeden Schlüssel
 \rightarrow Vergleichbarkeit

Angriff auf Ed25519 IIII







PATCH für SHA512 in WolfSSL

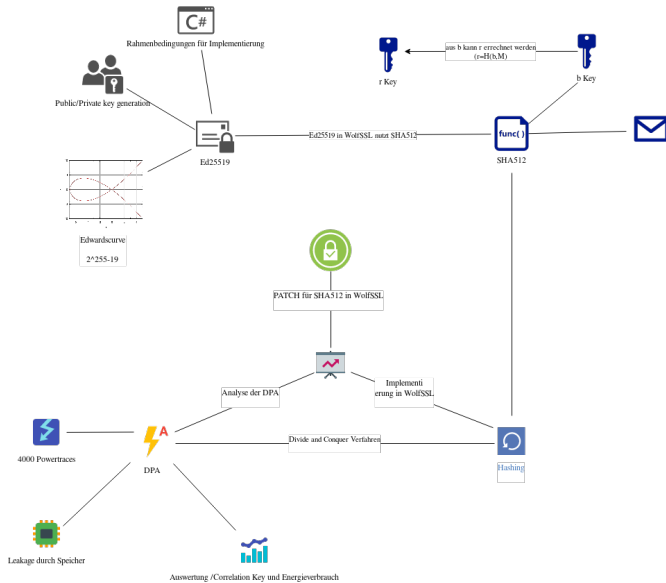


Analyse der DPA

Implementi
erung in WolfSSL

Verbesserung & Gegenmaßnahmen

- ▶ Schlüssel & Nachricht nicht gemeinsam in Kompressionsfunktion
- ▶ Padding bereits bei Schlüssel durchführen
 - ▶ Random Werte
- ▶ Vorteil
 - ▶ Verifikation Signatur bleibt gleich
 - ▶ Implementierung des SHA wird geändert
- ▶ Nachteil für IoT
 - ▶ Verlust der deterministischen Berechnung
 - ▶ Berechnungszeit steigt



Literatur

- Hühnlein, Detlef, and Ulrike Korte. 2006. *Grundlagen Der Elektronischen Signatur: Recht-Technik-Anwendung*. SecuMedia-Verlag.
- Katz, Jonathan, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 1996. *Handbook of Applied Cryptography*. CRC press.
- Susella, Ruggero. 2018. "Breaking Ed25519 in Wolfssl." In *Topics in Cryptology–Ct-Rsa 2018: The Cryptographers' Track at the Rsa Conference 2018, San Francisco, ca, Usa, April 16-20, 2018, Proceedings*, 10808:1. Springer.