Development of security-critical software with SPARK/Ada at secunet

Stefan Berghofer

30.5.2017

Agenda

1. Introduction
2. A Link between Why3 and Isabelle
3. Applications
4. Conclusion

Agenda

1. **Introduction**
2. A Link between Why3 and Isabelle
3. Applications
4. Conclusion

About secunet Security Networks AG

About secunet Security Networks AG

- One of Germany's leading providers of IT security

secunet

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:
  - **Public sector**

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:
  - **Public sector**
    Public Authorities, Homeland Security, Defence

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:
  - **Public sector**
    Public Authorities, Homeland Security, Defence
  - **Business sector**

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:
  - **Public sector**
    Public Authorities, Homeland Security, Defence
  - **Business sector**
    Automotive, Critical Infrastructures

About secunet Security Networks AG

- One of Germany's leading providers of IT security
- Security partner of the Federal Republic of Germany
- More than 400 employees
- Major shareholder is Giesecke & Devrient, Munich
- Business Units:
  - **Public sector**
    Public Authorities, Homeland Security, Defence
  - **Business sector**
    Automotive, Critical Infrastructures
- More details: `www.secunet.com`

Context

High-security VPN gateways and clients

Context

High-security VPN gateways and clients

- Process various categories of data with different classification

secunet

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

secunet

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small

secunet

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture
  - Few small, trusted components, e.g. encryption / decryption

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture
  - Few small, trusted components, e.g. encryption / decryption
  - Larger untrusted components, e.g. user sessions, device drivers, network protocol stack, ...

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture
  - Few small, trusted components, e.g. encryption / decryption
  - Larger untrusted components, e.g. user sessions, device drivers, network protocol stack, ...
- Separation kernel controls interaction of components

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture
  - Few small, trusted components, e.g. encryption / decryption
  - Larger untrusted components, e.g. user sessions, device drivers, network protocol stack, ...
- Separation kernel controls interaction of components
- Implement / verify trusted components using SPARK

Context

High-security VPN gateways and clients

- Process various categories of data with different classification
- Prevent unintended information flow between domains

Development approach

- Design goal: Keep trusted computing base small
- Component-based architecture
  - Few small, trusted components, e.g. encryption / decryption
  - Larger untrusted components, e.g. user sessions, device drivers, network protocol stack, ...
- Separation kernel controls interaction of components
- Implement / verify trusted components using SPARK
- Prove at least absence of runtime exceptions for all trusted components, for some also functional correctness

secunet

SPARK at secunet

2010 Implementation of components in SPARK 2005

SPARK at secunet

2010	Implementation of components in SPARK 2005
2011	Verification of components using SPARK and Isabelle

SPARK at secunet

2010 Implementation of components in SPARK 2005

2011 Verification of components using SPARK and Isabelle

2013 Isabelle driver for Why3

SPARK at secunet

2010 Implementation of components in SPARK 2005

2011 Verification of components using SPARK and Isabelle

2013 Isabelle driver for Why3

2014 Switch to SPARK 2014

SPARK at secunet

2010	Implementation of components in SPARK 2005
2011	Verification of components using SPARK and Isabelle
2013	Isabelle driver for Why3
2014	Switch to SPARK 2014 Introduction of Muen separation kernel

SPARK at secunet

2010 Implementation of components in SPARK 2005

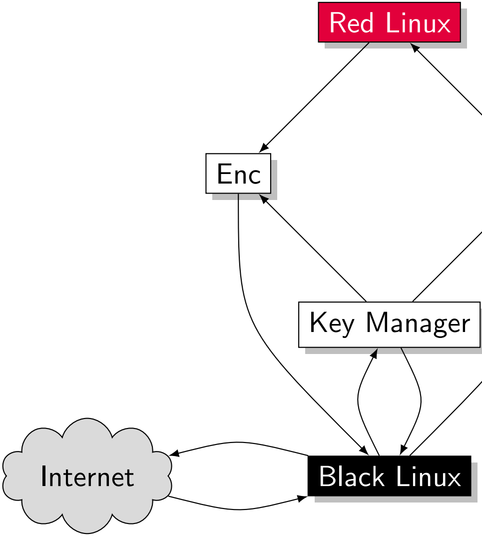
2011 Verification of components using SPARK and Isabelle

2013 Isabelle driver for Why3

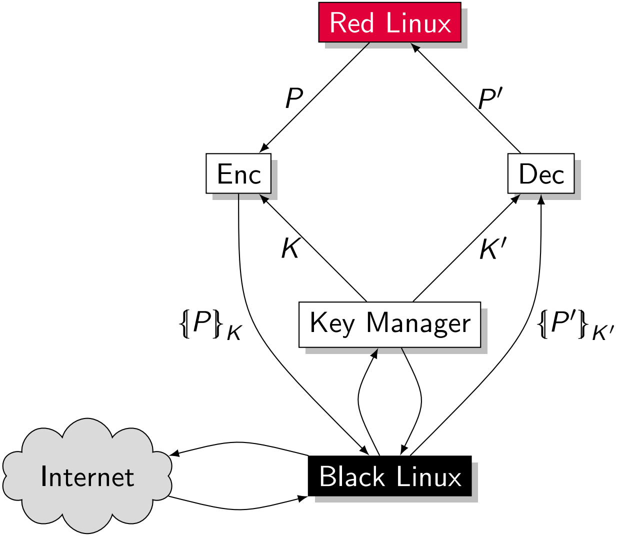
2014 Switch to SPARK 2014
Introduction of Muen separation kernel

2017 Proof of properties of Muen separation kernel (ongoing)

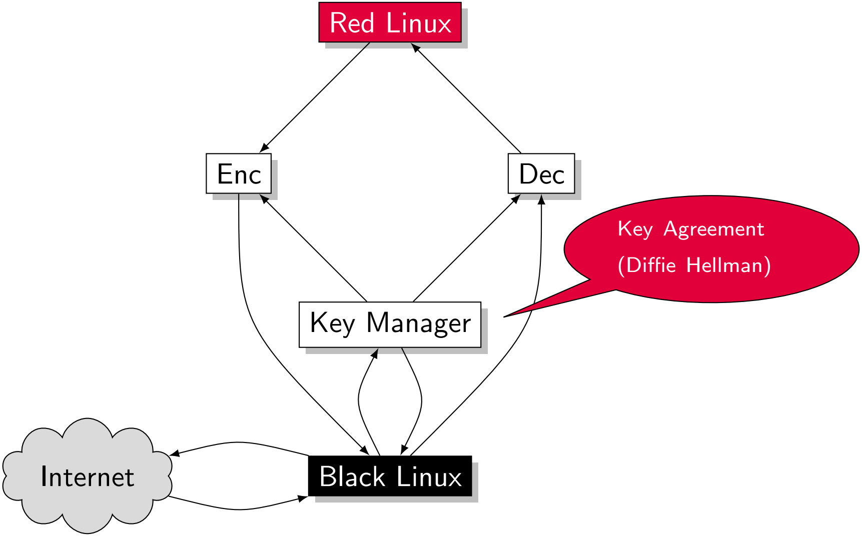
Component-Based Architecture



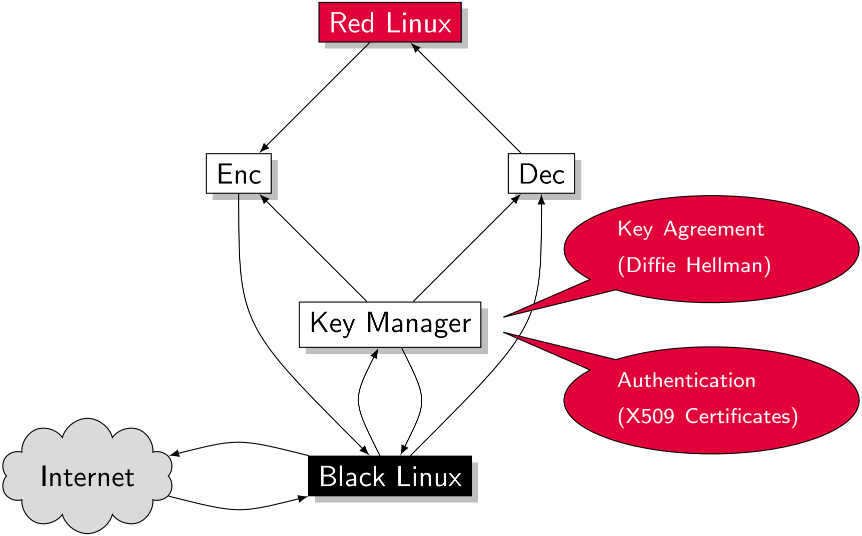
Component-Based Architecture



Component-Based Architecture



Component-Based Architecture



secunet

Verification Approaches

Verification Approaches

- Automatic

Verification Approaches

- Automatic
- Auto-Active

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers

secunet

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive

secunet

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive
  using Coq or Isabelle

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive
  using Coq or Isabelle

Why interactive verification?

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive
  using Coq or Isabelle

Why interactive verification?

- About 10% – 40% of the VCs generated from our codebase are not proved automatically by SMT solvers

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive
  using Coq or Isabelle

Why interactive verification?

- About 10% – 40% of the VCs generated from our codebase are not proved automatically by SMT solvers
- Lemma subprograms difficult to synthesize for complex proofs (tool support?)

Verification Approaches

- Automatic
- Auto-Active
  use lemma subprograms to help automatic provers
- Interactive
  using Coq or Isabelle

Why interactive verification?

- About 10% – 40% of the VCs generated from our codebase are not proved automatically by SMT solvers
- Lemma subprograms difficult to synthesize for complex proofs (tool support?)
- Complex external specifications can be linked to SPARK code using ghost functions

Agenda

1. Introduction
2. **A Link between Why3 and Isabelle**
3. Applications
4. Conclusion

Isabelle/HOL

Isabelle/HOL

- Interactive theorem prover

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA
- Specifications written in a functional programming language with logical operators

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA
- Specifications written in a functional programming language with logical operators
- Design philosophy

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA
- Specifications written in a functional programming language with logical operators
- Design philosophy
  - Inferences may only be performed by small kernel
    "LCF approach" [Robin Milner]

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA
- Specifications written in a functional programming language with logical operators
- Design philosophy
  - Inferences may only be performed by small kernel
    "LCF approach" [Robin Milner]
  - Definitional theory extension
    New concepts must be introduced using already existing and more primitive concepts.

Isabelle/HOL

- Interactive theorem prover
- Developed (since 1986) by Prof. Larry Paulson (Cambridge), Prof. Tobias Nipkow (Munich) and Dr. Markus Wenzel
- Widely used, e.g. in the L4 verified project at NICTA
- Specifications written in a functional programming language with logical operators
- Design philosophy
  - Inferences may only be performed by small kernel
    "LCF approach" [Robin Milner]
  - Definitional theory extension
    New concepts must be introduced using already existing and more primitive concepts.
- More information: `isabelle.in.tum.de`

Isabelle Driver for Why3

- Inspired by Coq driver

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

  Why3 Generates XML file with definitions and VCs

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

  Why3 Generates XML file with definitions and VCs

  Isabelle Parses XML file, performs definitions, manages VCs

secunet

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

  Why3 Generates XML file with definitions and VCs

  Isabelle Parses XML file, performs definitions, manages VCs
- Strict separation between generated and user-edited content

secunet

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

  Why3 Generates XML file with definitions and VCs

  Isabelle Parses XML file, performs definitions, manages VCs
- Strict separation between generated and user-edited content
- No fragile heuristics for parsing and interpreting edited parts

Isabelle Driver for Why3

- Inspired by Coq driver
- Uses data exchange format that is easy to generate and parse (XML, no Isabelle theory files)
- Consists of two parts

  Why3 Generates XML file with definitions and VCs

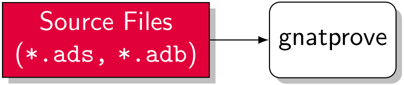
  Isabelle Parses XML file, performs definitions, manages VCs
- Strict separation between generated and user-edited content
- No fragile heuristics for parsing and interpreting edited parts
- System keeps track of proved / unproved VCs

SPARK Toolchain

Source Files
(*.ads, *.adb)

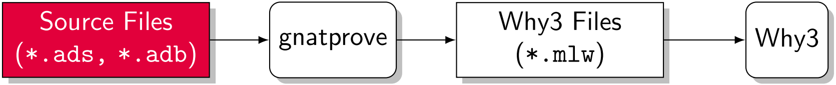
secunet

SPARK Toolchain

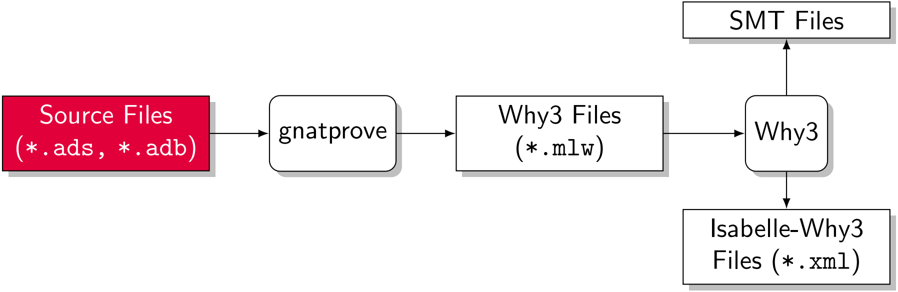


SPARK Toolchain

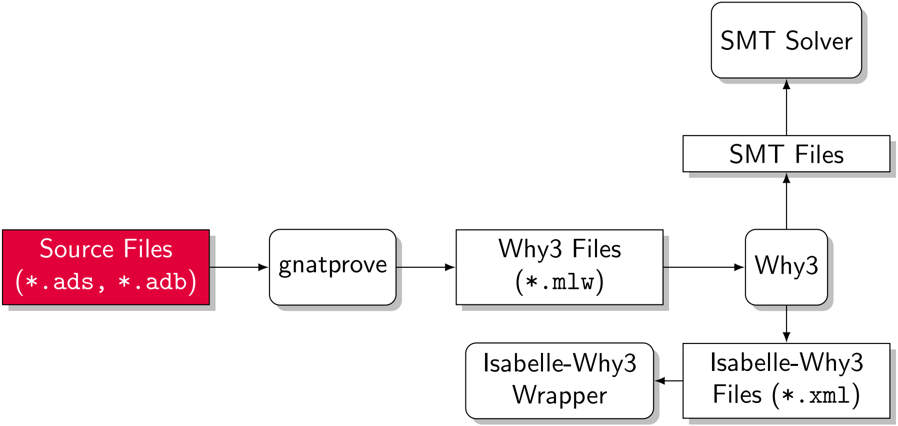
Spark Toolchain



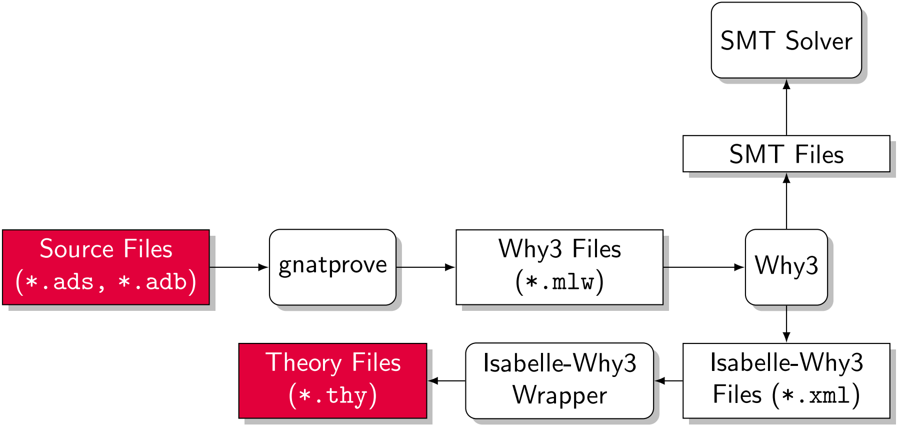
SPARK Toolchain



SPARK Toolchain

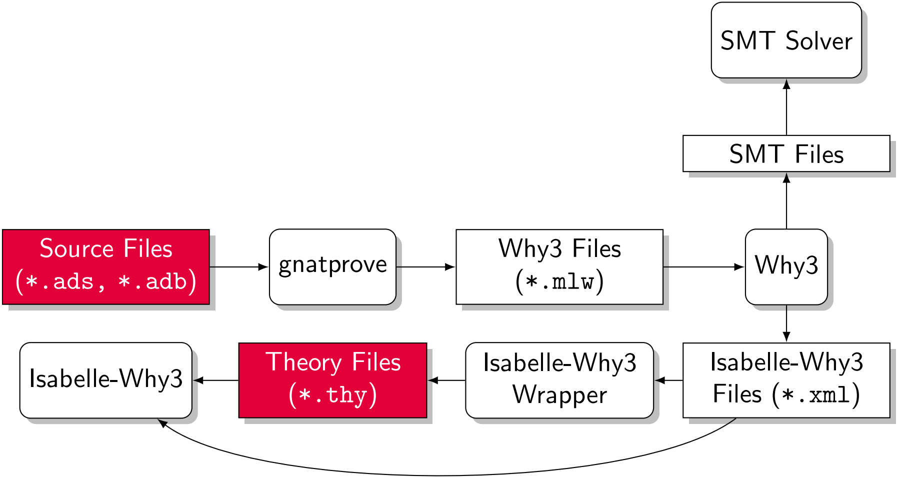


SPARK Toolchain



secunet

SPARK Toolchain

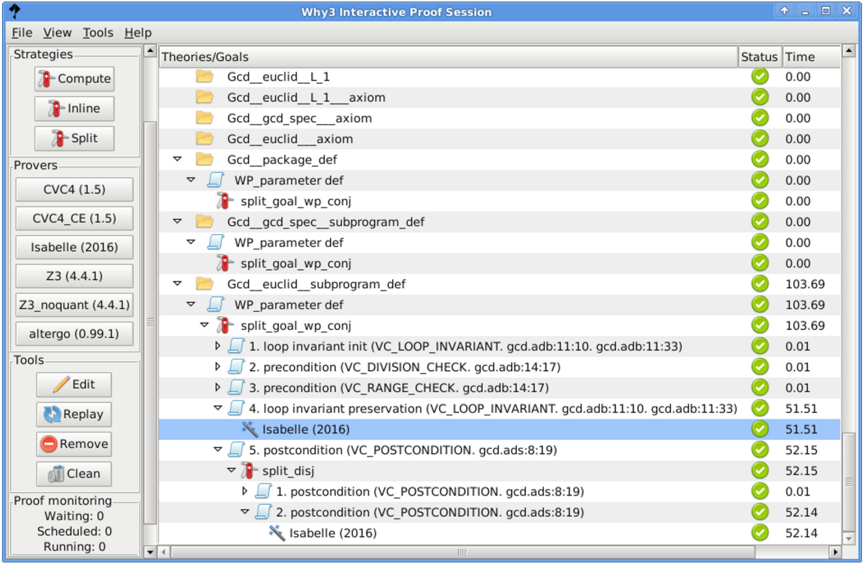


secunet

Example: Euclidean Algorithm

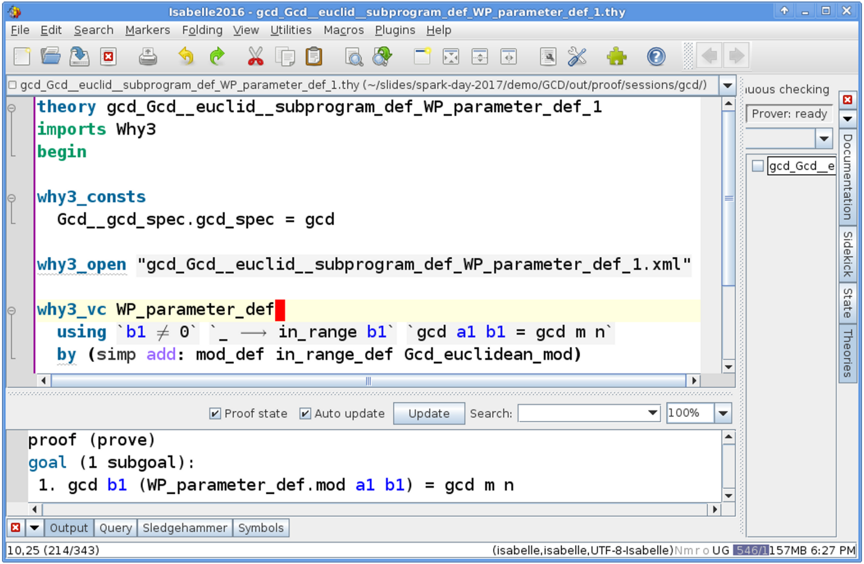
```
function GCD_Spec (M, N : Natural) return Natural
  with Ghost, Import;

function Euclid (M, N : Natural) return Natural
  with Post => Euclid'Result = GCD_Spec (M, N)
is
   A, B, R : Natural;
begin
   A := M; B := N;

   loop
      pragma Loop_Invariant (GCD_Spec (A, B) = GCD_Spec (M, N));
      exit when B = 0;
      R := A mod B;
      A := B; B := R;
   end loop;

   return A;
end Euclid;
```

Managing VCs using Why3 IDE



secunet

Proving VCs using Isabelle



secunet

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

secunet

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

why3_types Link Why3 types with Isabelle types

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

why3_types Link Why3 types with Isabelle types (works for uninterpreted or data types)

secunet

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

why3_types Link Why3 types with Isabelle types (works for uninterpreted or data types)

why3_defs Replace Why3 definitions by Isabelle definitions

secunet

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

why3_types Link Why3 types with Isabelle types (works for uninterpreted or data types)

why3_defs Replace Why3 definitions by Isabelle definitions

why3_thms Replace Why3 axioms by Isabelle theorems

Commands of Why3-Plugin for Isabelle

why3_open Load and parse VCs

why3_vc Start proof of VC

why3_end Close Why3 environment

why3_status Show VCs

why3_consts Link uninterpreted Why3 constants with Isabelle constants

why3_types Link Why3 types with Isabelle types (works for uninterpreted or data types)

why3_defs Replace Why3 definitions by Isabelle definitions

why3_thms Replace Why3 axioms by Isabelle theorems

why3_consts etc. can be viewed as light-weight on-the-fly variant of Why3's realizations that happen completely on the Isabelle side

Agenda

1. Introduction
2. A Link between Why3 and Isabelle
3. **Applications**
4. Conclusion

Supported algorithms

Supported algorithms

- Hash functions: (HMAC-)SHA-256/384/512

Supported algorithms

- Hash functions: (HMAC-)SHA-256/384/512
- Symmetric: AES-128/192/256

Supported algorithms

- Hash functions: (HMAC-)SHA-256/384/512
- Symmetric: AES-128/192/256
- Asymmetric: Elliptic Curves

libsparkcrypto – A Cryptographic Library for SPARK

Supported algorithms

- Hash functions: (HMAC-)SHA-256/384/512
- Symmetric: AES-128/192/256
- Asymmetric: Elliptic Curves

Open Source

`http://git.codelabs.ch/?p=spark-crypto.git`

secunet

Layers of Cryptographic Functions

Big numbers	Modular multiplication, addition, subtraction

Layers of Cryptographic Functions

Elliptic curves (basic)	Point addition and doubling
Big numbers	Modular multiplication, addition, subtraction

Layers of Cryptographic Functions

Elliptic curves (derived)	Scalar multiplication
Elliptic curves (basic)	Point addition and doubling
Big numbers	Modular multiplication, addition, subtraction

Layers of Cryptographic Functions

Security protocols	ECDSA, ECDH
Elliptic curves (derived)	Scalar multiplication
Elliptic curves (basic)	Point addition and doubling
Big numbers	Modular multiplication, addition, subtraction

Basic Declarations for Big Numbers

```
package Bignum
is

   function Base return Math_Int.Math_Int is (Math_Int.From_Word32 (2) ** 32)
     with Ghost;

   subtype Big_Int_Range is Natural range Natural'First .. Natural'Last - 1;

   type Big_Int is array (Big_Int_Range range <>) of Types.Word32;

   function Num_Of_Big_Int (A : Big_Int; F, L : Natural)
     return Math_Int.Math_Int
     with Ghost, Import, Global => null;

   function Num_Of_Boolean (B : Boolean) return Math_Int.Math_Int
     with Ghost, Import, Global => null;

   function Inverse (M, A : Math_Int.Math_Int) return Math_Int.Math_Int
     with Ghost, Import, Global => null;
                                              ...
end Bignum;
```

Formalization of Big Numbers in Isabelle

secunet

Formalization of Big Numbers in Isabelle

Abstraction function

$$\textit{num-of-big-int} :: (\textit{int} \Rightarrow \textit{int}) \Rightarrow \textit{int} \Rightarrow \textit{int} \Rightarrow \textit{int}$$
$$\textit{num-of-big-int}\ A\ k\ i = (\sum j = 0..{<}i.\ \textit{Base}^{j} * A\ (k + j))$$

secunet

Formalization of Big Numbers in Isabelle

Abstraction function

$$\mathit{num\text{-}of\text{-}big\text{-}int} :: (\mathit{int} \Rightarrow \mathit{int}) \Rightarrow \mathit{int} \Rightarrow \mathit{int} \Rightarrow \mathit{int}$$
$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ i = (\textstyle\sum j = 0..{<}i.\ \mathit{Base}^j * A\ (k + j))$$

Summation property

$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ (i + j) =$$
$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ i + \mathit{Base}^i * \mathit{num\text{-}of\text{-}big\text{-}int}\ A\ (k + i)\ j$$

Formalization of Big Numbers in Isabelle

Abstraction function

$$\mathit{num\text{-}of\text{-}big\text{-}int} :: (\mathit{int} \Rightarrow \mathit{int}) \Rightarrow \mathit{int} \Rightarrow \mathit{int} \Rightarrow \mathit{int}$$
$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ i = (\textstyle\sum j = 0..{<}i.\ \mathit{Base}^j * A\ (k + j))$$

Summation property

$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ (i + j) =$$
$$\mathit{num\text{-}of\text{-}big\text{-}int}\ A\ k\ i + \mathit{Base}^i * \mathit{num\text{-}of\text{-}big\text{-}int}\ A\ (k + i)\ j$$

Modular inverse

$$\mathit{minv} :: \mathit{int} \Rightarrow \mathit{int} \Rightarrow \mathit{int}$$
$$\mathit{coprime}\ x\ m \Longrightarrow 0 < x \Longrightarrow 1 < m \Longrightarrow x * \mathit{minv}\ m\ x\ \mathit{mod}\ m = 1$$

Montgomery Multiplication

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad \text{where } R = b^n$$

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad \text{where } R = b^n$$

Perform computations on numbers in Montgomery format

$$\widetilde{x} = x \cdot R \bmod m \qquad \text{(likewise for } \widetilde{y}\text{)}$$

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad\qquad \text{where } R = b^n$$

Perform computations on numbers in Montgomery format

$$\widetilde{x} = x \cdot R \bmod m \qquad\qquad (\text{likewise for } \widetilde{y})$$

Multiplication of numbers in Montgomery format

$$\widetilde{x} \otimes \widetilde{y} = x \cdot R \cdot y \cdot R \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m = \widetilde{x \cdot y}$$

secunet

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad \text{where } R = b^n$$

Perform computations on numbers in Montgomery format

$$\widetilde{x} = x \cdot R \bmod m \qquad \text{(likewise for } \widetilde{y}\text{)}$$

Multiplication of numbers in Montgomery format

$$\widetilde{x} \otimes \widetilde{y} = x \cdot R \cdot y \cdot R \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m = \widetilde{x \cdot y}$$

Conversion between standard and Montgomery format

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad\qquad \text{where } R = b^n$$

Perform computations on numbers in Montgomery format

$$\widetilde{x} = x \cdot R \bmod m \qquad\qquad \text{(likewise for } \widetilde{y}\text{)}$$

Multiplication of numbers in Montgomery format

$$\widetilde{x} \otimes \widetilde{y} = x \cdot R \cdot y \cdot R \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m = \widetilde{x \cdot y}$$

Conversion between standard and Montgomery format

$$x \otimes (R^2 \bmod m) = x \cdot R^2 \cdot R^{-1} \bmod m = x \cdot R \bmod m = \widetilde{x}$$

Montgomery Multiplication

Computes

$$x \otimes y = x \cdot y \cdot R^{-1} \bmod m \qquad \text{where } R = b^n$$

Perform computations on numbers in Montgomery format

$$\widetilde{x} = x \cdot R \bmod m \qquad \text{(likewise for } \widetilde{y}\text{)}$$

Multiplication of numbers in Montgomery format

$$\widetilde{x} \otimes \widetilde{y} = x \cdot R \cdot y \cdot R \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m = \widetilde{x \cdot y}$$

Conversion between standard and Montgomery format

$$x \otimes (R^2 \bmod m) = x \cdot R^2 \cdot R^{-1} \bmod m = x \cdot R \bmod m = \widetilde{x}$$

$$\widetilde{x} \otimes 1 = x \cdot R \cdot 1 \cdot R^{-1} \bmod m = x \bmod m$$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n - 1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n - 1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	$0 \quad +$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n - 1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

	0	+
$6 \cdot 789$	4734	=

$-7^{-1} \bmod 10 = 7$

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	$+$
$6 \cdot 789$	4734	$=$
	4734	$+$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=
	12630	/ 10 =

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=
	12630	/ 10 =
	1263	+

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	$+$
$6 \cdot 789$	4734	$=$
	4734	$+$
$8 \cdot 987$	7896	$=$
	12630	$/\ 10\ =$
	1263	$+$
$5 \cdot 789$	3945	$=$

Efficiently Computing $456 \cdot 789 \bmod 987$

$$
\begin{array}{l}
a \leftarrow 0 \\
\textbf{for } i = 0 \textbf{ to } n-1 \textbf{ do} \\
\quad u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b \\
\quad a \leftarrow (a + x_i \cdot y + u \cdot m)/b \\
\textbf{end for} \\
\textbf{if } a \geq m \textbf{ then} \\
\quad a \leftarrow a - m \\
\textbf{end if}
\end{array}
$$

$$-7^{-1} \bmod 10 = 7$$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=
	12630	/ 10 =
	1263	+
$5 \cdot 789$	3945	=
	5208	+

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	$+$
$6 \cdot 789$	4734	$=$
	4734	$+$
$8 \cdot 987$	7896	$=$
	12630	$/\ 10\ =$
	1263	$+$
$5 \cdot 789$	3945	$=$
	5208	$+$
$6 \cdot 987$	5922	$=$

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	$+$
$6 \cdot 789$	4734	$=$
	4734	$+$
$8 \cdot 987$	7896	$=$
	12630	$/\ 10\ =$
	1263	$+$
$5 \cdot 789$	3945	$=$
	5208	$+$
$6 \cdot 987$	5922	$=$
	11130	$/\ 10\ =$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	$+$
$6 \cdot 789$	4734	$=$
	4734	$+$
$8 \cdot 987$	7896	$=$
	12630	$/\ 10 =$
	1263	$+$
$5 \cdot 789$	3945	$=$
	5208	$+$
$6 \cdot 987$	5922	$=$
	11130	$/\ 10 =$
	1113	$+$

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=
	12630	/ 10 =
	1263	+
$5 \cdot 789$	3945	=
	5208	+
$6 \cdot 987$	5922	=
	11130	/ 10 =
	1113	+
$4 \cdot 789$	3156	=

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+	
$6 \cdot 789$	4734	=	
	4734	+	
$8 \cdot 987$	7896	=	
	12630	/ 10	=
	1263	+	
$5 \cdot 789$	3945	=	
	5208	+	
$6 \cdot 987$	5922	=	
	11130	/ 10	=
	1113	+	
$4 \cdot 789$	3156	=	
	4269	+	

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$$
\begin{aligned}
& a \leftarrow 0 \\
& \textbf{for } i = 0 \textbf{ to } n-1 \textbf{ do} \\
& \quad u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b \\
& \quad a \leftarrow (a + x_i \cdot y + u \cdot m)/b \\
& \textbf{end for} \\
& \textbf{if } a \geq m \textbf{ then} \\
& \quad a \leftarrow a - m \\
& \textbf{end if}
\end{aligned}
$$

$-7^{-1} \bmod 10 = 7$

$$
\begin{array}{r|rl}
 & 0 & + \\
6 \cdot 789 & 4734 & = \\
 & 4734 & + \\
8 \cdot 987 & 7896 & = \\
 & 12630 & /\ 10\ = \\
\hline
 & 1263 & + \\
5 \cdot 789 & 3945 & = \\
 & 5208 & + \\
6 \cdot 987 & 5922 & = \\
 & 11130 & /\ 10\ = \\
\hline
 & 1113 & + \\
4 \cdot 789 & 3156 & = \\
 & 4269 & + \\
3 \cdot 987 & 2961 & = \\
 & & \\
\hline
 & &
\end{array}
$$

Efficiently Computing $456 \cdot 789 \bmod 987$

$$
\begin{array}{l}
a \leftarrow 0 \\
\textbf{for } i = 0 \textbf{ to } n-1 \textbf{ do} \\
\quad u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b \\
\quad a \leftarrow (a + x_i \cdot y + u \cdot m)/b \\
\textbf{end for} \\
\textbf{if } a \geq m \textbf{ then} \\
\quad a \leftarrow a - m \\
\textbf{end if}
\end{array}
$$

$-7^{-1} \bmod 10 = 7$

$$
\begin{array}{r|rl}
 & 0 & + \\
6 \cdot 789 & 4734 & = \\
 & 4734 & + \\
8 \cdot 987 & 7896 & = \\
 & 12630 & /\ 10\ = \\
\hline
 & 1263 & + \\
5 \cdot 789 & 3945 & = \\
 & 5208 & + \\
6 \cdot 987 & 5922 & = \\
 & 11130 & /\ 10\ = \\
\hline
 & 1113 & + \\
4 \cdot 789 & 3156 & = \\
 & 4269 & + \\
3 \cdot 987 & 2961 & = \\
 & 7230 & /\ 10\ = \\
\hline
 & &
\end{array}
$$

secunet

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n - 1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

	0	+
$6 \cdot 789$	4734	=
	4734	+
$8 \cdot 987$	7896	=
	12630	/ 10 =
	1263	+
$5 \cdot 789$	3945	=
	5208	+
$6 \cdot 987$	5922	=
	11130	/ 10 =
	1113	+
$4 \cdot 789$	3156	=
	4269	+
$3 \cdot 987$	2961	=
	7230	/ 10 =
	723	

Efficiently Computing $456 \cdot 789 \bmod 987$

$a \leftarrow 0$
for $i = 0$ **to** $n-1$ **do**
  $u \leftarrow (a_0 + x_i \cdot y_0) \cdot -m_0^{-1} \bmod b$
  $a \leftarrow (a + x_i \cdot y + u \cdot m)/b$
end for
if $a \geq m$ **then**
  $a \leftarrow a - m$
end if

$-7^{-1} \bmod 10 = 7$

$723 \cdot 1000 \bmod 987 = 516 =$
$456 \cdot 789 \bmod 987$

	0	+	
$6 \cdot 789$	4734	=	
	4734	+	
$8 \cdot 987$	7896	=	
	12630	/ 10	=
	1263	+	
$5 \cdot 789$	3945	=	
	5208	+	
$6 \cdot 987$	5922	=	
	11130	/ 10	=
	1113	+	
$4 \cdot 789$	3156	=	
	4269	+	
$3 \cdot 987$	2961	=	
	7230	/ 10	=
	723		

Montgomery Multiplication in SPARK

```
for I in Natural range A_First .. A_Last
loop
   Carry1 := 0; Carry2 := 0;
   XI := X (X_First + (I - A_First));
   U := (A (A_First) + XI * Y (Y_First)) * M_Inv;
   Single_Add_Mult_Mult
     (A (A_First), XI, Y (Y_First),
      M (M_First), U, Carry1, Carry2);
   Add_Mult_Mult
     (A, A_First, A_Last - 1,
      Y, Y_First + 1, M, M_First + 1,
      XI, U, Carry1, Carry2);
   A (A_Last) := A_MSW + Carry1;
   A_MSW := Carry2 + Word_Of_Boolean (A (A_Last) < Carry1);
end loop;
```

Specification of Montgomery Multiplication

Preconditions

```
Num_Of_Big_Int (Y, Y_First, A_Last - A_First + 1) <
Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 < Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 + M_Inv * M (M_First) = 0
```

Specification of Montgomery Multiplication

Preconditions

```
Num_Of_Big_Int (Y, Y_First, A_Last - A_First + 1) <
Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 < Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 + M_Inv * M (M_First) = 0
```

Postcondition

```
Num_Of_Big_Int (A, A_First, A_Last - A_First + 1) =
(Num_Of_Big_Int (X, X_First, A_Last - A_First + 1) *
 Num_Of_Big_Int (Y, Y_First, A_Last - A_First + 1) *
 Inverse (Num_Of_Big_Int (M, M_First, A_Last - A_First + 1),
   Base) ** (A_Last - A_First + 1)) mod
Num_Of_Big_Int (M, M_First, A_Last - A_First + 1)
```

Specification of Montgomery Multiplication

Preconditions

```
Num_Of_Big_Int (Y, Y_First, A_Last - A_First + 1) <
Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 < Num_Of_Big_Int (M, M_First, A_Last - A_First + 1) and
1 + M_Inv * M (M_First) = 0
```

Postcondition

```
Num_Of_Big_Int (A, A_First, A_Last - A_First + 1) =
(Num_Of_Big_Int (X, X_First, A_Last - A_First + 1) *
 Num_Of_Big_Int (Y, Y_First, A_Last - A_First + 1) *
 Inverse (Num_Of_Big_Int (M, M_First, A_Last - A_First + 1),
   Base) ** (A_Last - A_First + 1)) mod
Num_Of_Big_Int (M, M_First, A_Last - A_First + 1)
```

In mathematical notation...

$$a = x \cdot y \cdot b^{-n} \bmod m$$

Elliptic Curves

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)

secunet

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective (x, y, z)

secunet

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective $(x, y, z) \quad \longmapsto \quad (x/z, y/z)$

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective $(x, y, z) \longmapsto (x/z, y/z)$
    point addition does not require computation of inverse

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective $(x, y, z) \quad \longmapsto \quad (x/z, y/z)$
    point addition does not require computation of inverse
- Abstract properties formalized in Isabelle

Elliptic Curves

- Applications: ECDH (key agreement) and
  ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective $(x, y, z) \quad \longmapsto \quad (x/z, y/z)$
    point addition does not require computation of inverse
- Abstract properties formalized in Isabelle
  based on Coq formalization by Laurent Théry

Elliptic Curves

- Applications: ECDH (key agreement) and ECDSA (authentication)
- Curves in Weierstrass form over fields of characteristic > 2
- Primitive operation: point addition
- Points on curve form a group wrt. point addition
- Coordinate systems:
  - Affine (x, y)
  - Projective $(x, y, z) \longmapsto (x/z, y/z)$
    point addition does not require computation of inverse
- Abstract properties formalized in Isabelle based on Coq formalization by Laurent Théry
- Proved correspondence of SPARK implementation with abstract specification

Elliptic Curves – Abstract Specification

$$\textbf{datatype}\ 'a\ point = Infinity \mid Point\ 'a\ 'a$$

$$\textbf{locale}\ \mathit{ell\text{-}field} = \mathit{field} +$$
$$\quad \textbf{assumes}\ \mathit{two\text{-}not\text{-}zero}\text{: } \ll 2 \gg \neq \mathbf{0}$$

$$\textbf{definition}\ \mathit{nonsingular} :: 'a \Rightarrow 'a \Rightarrow \mathit{bool}\ \textbf{where}$$
$$\quad \mathit{nonsingular}\ a\ b = (\ll 4 \gg \otimes a \uparrow 3 \oplus \ll 27 \gg \otimes b \uparrow 2 \neq \mathbf{0})$$

$$\textbf{definition}\ \mathit{on\text{-}curve} :: 'a \Rightarrow 'a \Rightarrow 'a\ point \Rightarrow \mathit{bool}\ \textbf{where}$$
$$\quad \mathit{on\text{-}curve}\ a\ b\ p = (\mathit{case}\ p\ \mathit{of}$$
$$\qquad Infinity \Rightarrow True$$
$$\qquad \mid Point\ x\ y \Rightarrow x \in \mathit{carrier}\ R \wedge y \in \mathit{carrier}\ R \wedge$$
$$\qquad\quad y \uparrow 2 = x \uparrow 3 \oplus a \otimes x \oplus b)$$

Point Addition

$$
\begin{array}{l}
\textbf{definition}\ add :: {}'a \Rightarrow {}'a\ point \Rightarrow {}'a\ point \Rightarrow {}'a\ point\ \textbf{where} \\
\quad add\ a\ p_1\ p_2 = (case\ p_1\ of \\
\qquad\quad Infinity \Rightarrow p_2 \\
\qquad \mid Point\ x_1\ y_1 \Rightarrow (case\ p_2\ of \\
\qquad\qquad Infinity \Rightarrow p_1 \\
\qquad\quad \mid Point\ x_2\ y_2 \Rightarrow \\
\qquad\qquad\quad if\ x_1 = x_2\ then \\
\qquad\qquad\qquad if\ y_1 = \ominus\ y_2\ then\ Infinity \\
\qquad\qquad\qquad else\ let\ l = (\ll 3 \gg \otimes x_1 \uparrow 2 \oplus a) \oslash (\ll 2 \gg \otimes y_1); \\
\qquad\qquad\qquad\qquad\quad x_3 = l \uparrow 2 \ominus \ll 2 \gg \otimes x_1 \\
\qquad\qquad\qquad\qquad in\ Point\ x_3\ (\ominus\ y_1 \ominus l \otimes (x_3 \ominus x_1)) \\
\qquad\qquad\quad else\ let\ l = (y_2 \ominus y_1) \oslash (x_2 \ominus x_1); \\
\qquad\qquad\qquad\qquad x_3 = l \uparrow 2 \ominus x_1 \ominus x_2 \\
\qquad\qquad\qquad in\ Point\ x_3\ (\ominus\ y_1 \ominus l \otimes (x_3 \ominus x_1))))
\end{array}
$$

Point Addition – Properties

lemma *add-closed*:
assumes $a \in carrier\ R$ **and** $b \in carrier\ R$
and $on\text{-}curve\ a\ b\ p_1$ **and** $on\text{-}curve\ a\ b\ p_2$
shows $on\text{-}curve\ a\ b\ (add\ a\ p_1\ p_2)$

Point Addition – Properties

lemma *add-closed*:
  assumes $a \in carrier\ R$ **and** $b \in carrier\ R$
  and $on\text{-}curve\ a\ b\ p_1$ **and** $on\text{-}curve\ a\ b\ p_2$
  shows $on\text{-}curve\ a\ b\ (add\ a\ p_1\ p_2)$

lemma *add-comm*:
  assumes $a \in carrier\ R$ **and** $b \in carrier\ R$
  and $on\text{-}curve\ a\ b\ p_1$ **and** $on\text{-}curve\ a\ b\ p_2$
  shows $add\ a\ p_1\ p_2 = add\ a\ p_2\ p_1$

Point Addition – Properties

lemma *add-closed*:
assumes $a \in carrier\ R$ **and** $b \in carrier\ R$
and $on\text{-}curve\ a\ b\ p_1$ **and** $on\text{-}curve\ a\ b\ p_2$
shows $on\text{-}curve\ a\ b\ (add\ a\ p_1\ p_2)$

lemma *add-comm*:
assumes $a \in carrier\ R$ **and** $b \in carrier\ R$
and $on\text{-}curve\ a\ b\ p_1$ **and** $on\text{-}curve\ a\ b\ p_2$
shows $add\ a\ p_1\ p_2 = add\ a\ p_2\ p_1$

lemma *add-assoc*:
assumes a: $a \in carrier\ R$ **and** b: $b \in carrier\ R$
and ab: $nonsingular\ a\ b$
and p_1: $on\text{-}curve\ a\ b\ p_1$ **and** p_2: $on\text{-}curve\ a\ b\ p_2$
and p_3: $on\text{-}curve\ a\ b\ p_3$
shows $add\ a\ p_1\ (add\ a\ p_2\ p_3) = add\ a\ (add\ a\ p_1\ p_2)\ p_3$

Projective Coordinates

type-synonym $'a\ ppoint = {'a} \times {'a} \times {'a}$

definition (**in** *field*) *make-affine* :: $'a\ ppoint \Rightarrow {'a}\ point$ **where**
make-affine $p =$
$(let\ (x, y, z) = p$
$in\ if\ z = \mathbf{0}\ then\ Infinity\ else\ Point\ (x \oslash z)\ (y \oslash z))$

lemma (**in** *ell-field*) *padd-correct*:
assumes a: $a \in carrier\ R$ **and** b: $b \in carrier\ R$
and p_1: $on\text{-}curvep\ a\ b\ p_1$ **and** p_2: $on\text{-}curvep\ a\ b\ p_2$
shows $make\text{-}affine\ (padd\ a\ p_1\ p_2) =$
$add\ a\ (make\text{-}affine\ p_1)\ (make\text{-}affine\ p_2)$

Projective Coordinates – Equality

definition (**in** *cring*) *proj-eq* :: $'a$ *ppoint* $\Rightarrow$ $'a$ *ppoint* $\Rightarrow$ *bool*
where

$$
\begin{aligned}
&\textit{proj-eq} = (\lambda(x_1, y_1, z_1)\ (x_2, y_2, z_2).\\
&\quad (z_1 = \mathbf{0}) = (z_2 = \mathbf{0}) \land\\
&\quad x_1 \otimes z_2 = x_2 \otimes z_1 \land y_1 \otimes z_2 = y_2 \otimes z_1)
\end{aligned}
$$

lemma (**in** *field*) *make-affine-proj-eq-iff*:

$$
\begin{aligned}
&\textit{in-carrierp}\ p \Longrightarrow \textit{in-carrierp}\ p' \Longrightarrow\\
&\ \textit{proj-eq}\ p\ p' = (\textit{make-affine}\ p = \textit{make-affine}\ p')
\end{aligned}
$$

Specification of Point Addition – SPARK Part

```
function Point_Add_Spec
  (M, A, X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3 : Math_Int.Math_Int)
  return Boolean
  with Ghost, Import, Global => null;

procedure Point_Add
  (X1, Y1, Z1 : in     Bignum.Big_Int;
   X2, Y2, Z2 : in     Bignum.Big_Int;
   X3, Y3, Z3 :    out Bignum.Big_Int;
   ...
   A          : in     Bignum.Big_Int;
   M          : in     Bignum.Big_Int;
   M_Inv      : in     Types.Word32)
  with
    Depends => ...
    Pre => ...
    Post =>
      Point_Add_Spec
        (Bignum.Num_Of_Big_Int (M, M_First, X1_Last - X1_First + 1),
         ...);
```

Specification of Point Addition – Isabelle Part

definition *point-add-spec* :: *math-int* ⇒ *math-int* ⇒
math-int ⇒ *math-int* ⇒ *math-int* ⇒ *math-int* ⇒ *math-int* ⇒
math-int ⇒ *math-int* ⇒ *math-int* ⇒ *math-int* ⇒ *bool*
where
point-add-spec m a x_1 y_1 z_1 x_2 y_2 z_2 x_3 y_3 z_3 =
(*let r* = *residue-ring* (*int-of-math-int m*);
a' = *int-of-math-int a mod int-of-math-int m*
in cring.proj-eq r
(*cring.padd r a'*
(*int-of-math-int* x_1, *int-of-math-int* y_1, *int-of-math-int* z_1)
(*int-of-math-int* x_2, *int-of-math-int* y_2, *int-of-math-int* z_2))
(*int-of-math-int* x_3, *int-of-math-int* y_3, *int-of-math-int* z_3))

why3-consts
Lsc--ec--point-add-spec.point-add-spec = *point-add-spec*

Agenda

1. Introduction
2. A Link between Why3 and Isabelle
3. Applications
4. **Conclusion**

Achievements

- Correctness of complex mathematical algorithms can be proved using SPARK

secunet

Achievements

- Correctness of complex mathematical algorithms can be proved using SPARK
- Link with interactive prover allows to prove advanced properties that are beyond reach of automatic provers

Achievements

- Correctness of complex mathematical algorithms can be proved using SPARK
- Link with interactive prover allows to prove advanced properties that are beyond reach of automatic provers
- Behaviour of programs can be specified in an abstract way

Achievements

- Correctness of complex mathematical algorithms can be proved using SPARK
- Link with interactive prover allows to prove advanced properties that are beyond reach of automatic provers
- Behaviour of programs can be specified in an abstract way
- SPARK implementation can be shown to correspond to abstract specification

Challenges

- Why3 model not really suitable for human consumption

Challenges

- Why3 model not really suitable for human consumption
- Translation to Why3 introduces many axioms
  (must be realized in Isabelle)

secunet

Challenges

- Why3 model not really suitable for human consumption
- Translation to Why3 introduces many axioms
  (must be realized in Isabelle)
- Why3 session file not perfectly suitable for interactive provers

Challenges

- Why3 model not really suitable for human consumption
- Translation to Why3 introduces many axioms
  (must be realized in Isabelle)
- Why3 session file not perfectly suitable for interactive provers
  e.g. goal matching algorithm sometimes gets confused by
  complex control flow

secunet

Ongoing and Future Work

- Verification of Muen separation kernel

secunet

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour

secunet

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables
  - Non-interference properties

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables
  - Non-interference properties
  - Abstract Isabelle model

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables
  - Non-interference properties
  - Abstract Isabelle model
- Improvement of Why3 plugin for Isabelle

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables
  - Non-interference properties
  - Abstract Isabelle model
- Improvement of Why3 plugin for Isabelle
  Automatic realization of axioms generated for SPARK types

secunet

Ongoing and Future Work

- Verification of Muen separation kernel
  - Runtime behaviour
    e.g. correct saving / restoring of subject states
  - Memory layout
    e.g. correct construction of page tables
  - Non-interference properties
  - Abstract Isabelle model
- Improvement of Why3 plugin for Isabelle
  Automatic realization of axioms generated for SPARK types
- Continued verification of security-critical components and protocols

secunet

secunet

secunet Security Networks AG

Stefan Berghofer

Kurfürstenstraße 58
45138 Essen
Tel.: +49-201-5454-3606
Fax: +49-201-5454-1323
stefan.berghofer@secunet.com
www.secunet.com