Securing Public-key Cryptography on the Android Platform

Anna Gommerstadt

Harvard Programming Languages Seminar 4/17/13

Me

- I’m a Senior concentrating in Computer Science and Math at Harvard College
- This work is my senior thesis, advised by Stephen Chong and Aslan Askarov
- This is my first PL Seminar talk!

Motivation

- The Android platform is notoriously insecure
- There are many different classes of bugs present on the platform
- A recent report found that one of the leading causes of errors is misused cryptography (40% of errors)

Cryptographic Errors

- Most developers rely on third party libraries for encryption and decryption
- We assume that these libraries provide correct implementations of common algorithms (RSA, etc)
- We are concerned with misuses of correctly implemented cryptographic protocols

What Kinds of Bugs?

- Mainly hardcoded cryptographic keys
  - This is a huge problem because Android applications are trivial to decompile, so a hardcoded key is easy to retrieve
- Also outputs of secure keys to public insecure channels

Background

- Aslan et al (2008) defined a semantic security guarantee and an enforcement mechanism for private-key cryptography
- Other work has been done on showing probabilistic guarantees of programs that use cryptography
  - We show a possibilistic guarantee which is simpler, but easier to compose with other policies

Approach

- Theory
  - Defining a semantic security condition for public-key cryptography
  - Developing an enforcement mechanism (a type system)
- Implementation
  - Developing Cryptflow, an information flow analysis that detects misuses of cryptography

Security Condition

- Based on the (awesome!) work by Aslan et al (2008)
- Standard noninterference does not work for public key cryptography because the public ciphertext is influenced by the high-security plaintext
- So we use possibilistic noninterference!

Possibilistic Noninterference

- A modified notion of traditional noninterference
- Basically, we want the ciphertext to possibly be any value
- Then, a change in the high-security plaintext does not affect the low-security ciphertext!

Encryption Semantics

- Encryption is nondeterministic
  - Encrypting a plaintext with a specific key can generate a lot of possible ciphertexts.
- Decryption is deterministic
  - A ciphertext and a key have one possible decryption
- Examples: El Gamal, nondeterministic variant of RSA

The Type System

- Using a standard imperative language with encryption, decryption and key generation commands
- Most of the typing is standard, except for input, output, encryption, decryption and key generation

Cool things in the Type System!

- We provide input and output channels with a lot of structure
  - Can output public and private keys on dedicated channels without causing massive security vulnerabilities
- Full details of the type system in the paper (I decided to spare you, it's a nice day)

Cryptflow

- Built on top of the Polyglot and Objanal frameworks
- Performs a flow sensitive information flow analysis of Java code
- Computes constraints on the levels of variables and tries to solve them
  - Solution: Flows are secure!
  - No Solution: Insecure flow!

Cryptflow Currently

- Can analyze snippets of code that misuse cryptography and identify simple vulnerabilities:
  - Outputs of private keys to System.out
  - Hardcoded private keys
- Still in prototype form (but I’m working on it!)

Future Work

- Working on having Cryptflow analyze actual Android applications to analyze misused cryptography
  - Analyzing the source Java code to find illegal flows of information having to do with cryptography
- Saving the world one cryptographic vulnerability at a time!

More Details

- Talk to me!
- My thesis is online on my/SEAS’ website
- This material will be in a PLDI poster this summer!