Introduction

- **ZK Proofs are Basic Crypto Primitives**
  - Used in identification schemes, group signatures, secure multiparty computation, ...
- **First ZK Protocols Deployed in Practice**
  - Direct anonymous attestation (DAA) in TPM chips
  - Anonymous credential systems (IBM identity mixer)
- **Based on Efficient Sigma-Protocols**
  - Protocol composition (AND, OR, ...) 
  - Generics transformations to non-interactive ZK (NIZK), ...
- **Design & Implementation “by Hand”**
  - Time-consuming, error-prone, ...

Challenges

- **Protocol Design**
  - Choose suitable proof techniques and parameters
- **Implementation Efficiency**
  - From semantic proof goal to implementation
  - Skill gap between cryptographer & programmers
- **Code Efficiency & Security**
  - Optimize resources at protocol & code level
  - Buffer overflows, SW side-channels, ...

Automatic Generation of Sound ZK Protocols

High Level ZK-PoK Language

- Relation to prove, security level, optimization constraints, ...
- Automatically find most efficient proof technique.

Protocol Specification Language

- Multiple types of sigma-protocols, groups, arbitrary compositions (AND, OR, ...), ...
- Automatic protocol composition and choice of parameters.

Protocol Implementation Language

- Algorithms, operations, messages, ...
- Easy to change from GMP to other libraries.

\[ \Sigma_{\text{exp}} \text{ – an Efficient Unconditionally Portable Protocol} \]

Applications

- Proofs in hidden-order groups:
  - Interval proofs
    - Bougout (Eurocrypt 00)
    - Lipmaa (ePrint 01)
  - Anonymous credential systems
    - Camenisch, Lysianskaya (SCN 02)
    - Camenisch, Van Herreweghen (ACM CCS 02)
  - E-Cash
    - Camenisch, Hohenberger, Lysianskaya (Eurocrypt 05)

High-Level Description of \( \Sigma_{\text{exp}} \)

\[
ZPK \left[ (x_1, ..., x_m) : y = \phi(x_1, ..., x_m) \right]
\]

Auxiliary String

- \( n : \text{RSA-Modulus} \)
  - \( g \in \mathbb{Z}_n^* \)
  - \( g_1, ..., g_m \in C \)

Prover

- \( t := \phi(r_1, ..., r_m) \)
- \( t' := g_1^{r_1} ... g_m^{r_m} \)
- \( y' := g_1^{y_1} ... g_m^{y_m} \)

Verifier

- \( c \in C \)
- \( s := r + c \)
- \( t y' = \phi(s_1, ..., s_m) \)
- \( t' y'' = g_1^{s_1} ... g_m^{s_m} \)

Related Work

- Bangertner (PhD-Thesis 05)
- Bangertner, Camenisch, Maurer (PKC 05)
- Camenisch, Kiayias, Yung (Eurocrypt 09)

Advantage

\( \Sigma_{\text{exp}} \text{ has 3 instead of 6 moves:} \)

- Higher Efficiency
- Efficient transformations to non-interactive ZK
- Concurrent ZK
- Generic protocol composition

EU FP7 Project Computer Aided Cryptography Engineering (CACE) - http://www.cace-project.eu