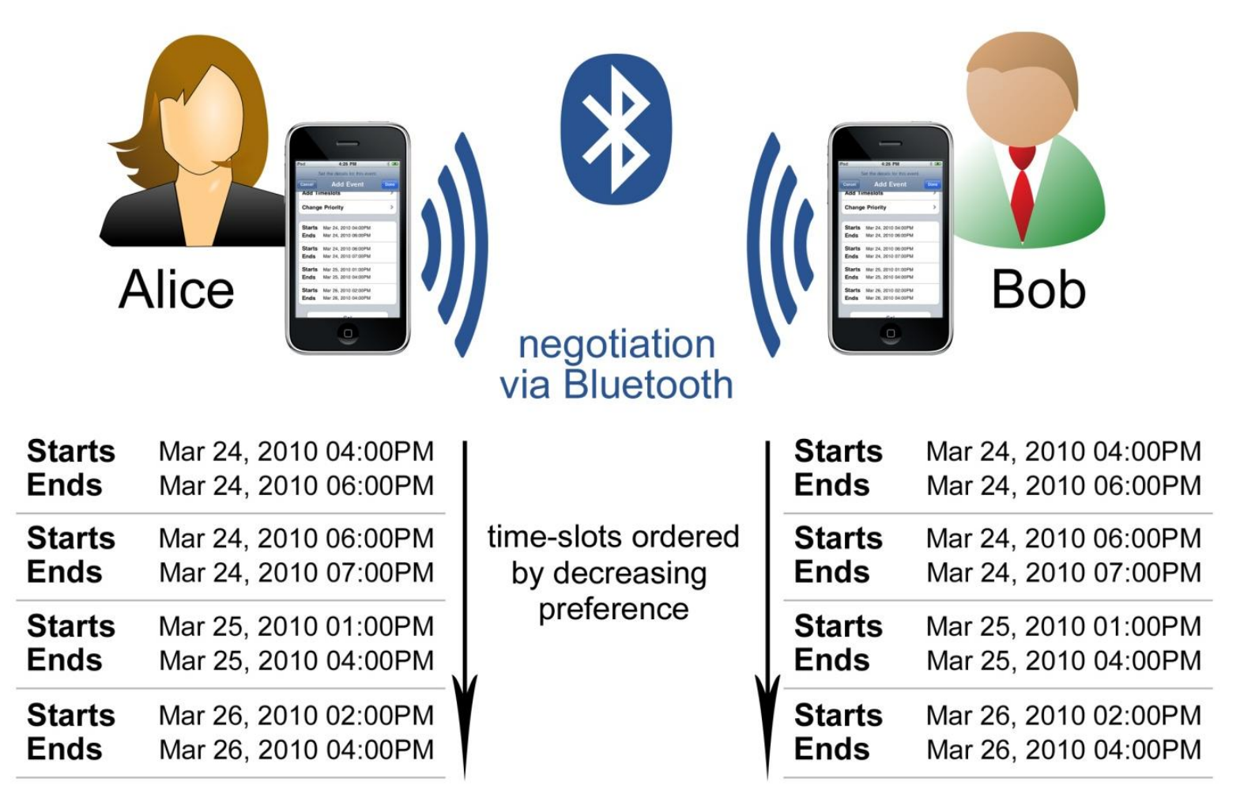
# appoint – A Distributed Privacy-Preserving iPhone Application

Daniel A. Mayer, Susanne Wetzel (Stevens Institute of Technology)
Dominik Teubert, Georg Neugebauer, Ulrike Meyer (RWTH Aachen, Germany)

## Introduction

### Policy Reconciliation

- To enable collaboration of two parties it is necessary that they agree on a common rule for this collaboration.
- The process of negotiating this common rule is called a reconciliation protocol.


Alice
Bob
negotiation via Bluetooth
time-slots ordered by decreasing preference

| | |
|---|---|
| Starts | Mar 24, 2010 04:00PM |
| Ends | Mar 24, 2010 06:00PM |
| Starts | Mar 24, 2010 06:00PM |
| Ends | Mar 24, 2010 07:00PM |
| Starts | Mar 25, 2010 01:00PM |
| Ends | Mar 25, 2010 04:00PM |
| Starts | Mar 26, 2010 02:00PM |
| Ends | Mar 26, 2010 04:00PM |

### The Challenge

- Current solutions do not take privacy concerns into account.
- Often, at least one of the parties must disclose all its parameter choices.

### Example: Scheduling an appointment using the iPhone

- Parties: two iPhone users
- Policies: A set of possible time-slots ordered by preference.
- Goal: Determining a time-slot that works best for both parties without revealing possible time-slots.

## Our PPPR Protocols

New protocols which meet the privacy requirements of the organizations and allow parties to find a common policy rule which optimizes their individual preferences.

### Our Protocols

- Common Policy (PCP) and Common Policy Cardinality (PC$^2$)
  - A and B learn nothing about each other's private policies but
    - which policy rules they have in common: $P_A \cap P_B$
    - how many policies they have in common: $|P_A \cap P_B|$
- Preference-Maximizing Protocol 3PR$^C$
  - Privacy-Preserving Preference-Maximizing Composition Scheme.
  - Upon completion of 3PR$^C$, A and B learn nothing about each other's policies but
    - one common, preference-maximizing policy rule

## Used Methodology

- The new protocols extend set intersection protocols introduced by Freedman et. al.[2]
- In order to find a match in their policies Alice and Bob use the following basic principle:
  1. Alice encodes the rules of her policy as roots of a polynomial.
  2. She then encrypts its coefficients and sends them to Bob.
  3. Bob evaluates the encrypted polynomial using each of his rules and sends the result back.
  4. Alice then decrypts the result and can determine whether a match was found (in this case the polynomial would evaluate to zero).

## Ongoing and Future Work

### Protocols

- Design protocols for malicious model.
- Extend protocols to multi-party case.
- Investigate alternate policy representations.

### iPhone App

- Extend appoint to multi-party case.
- More efficient policy representations.
- Handle partial matches.

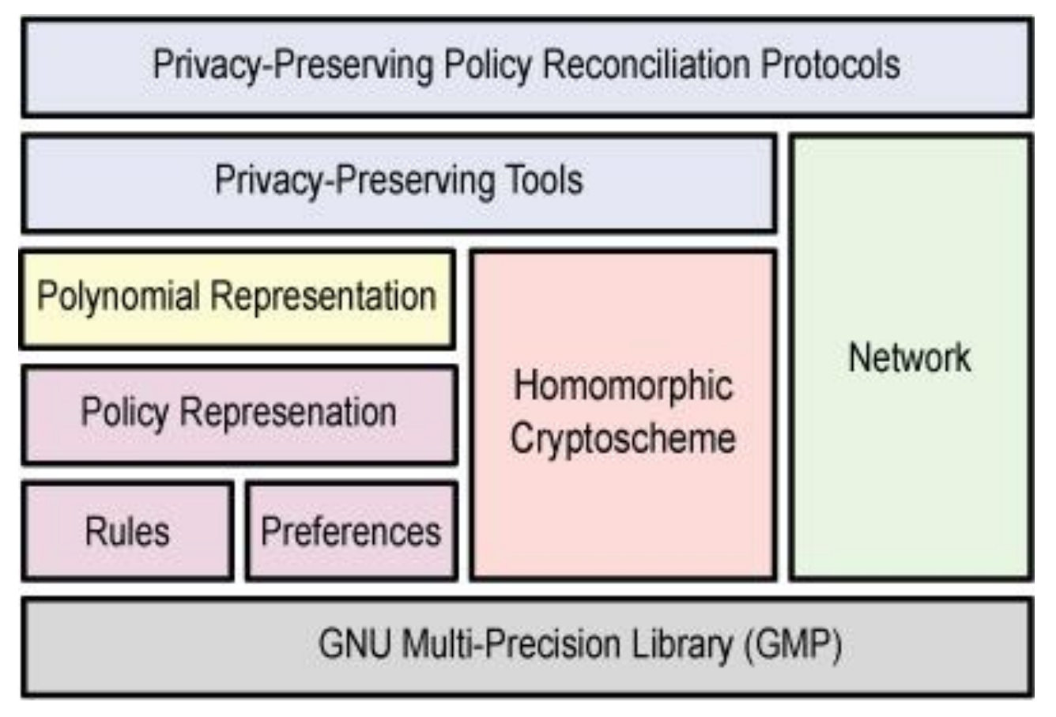
## References

[1] Meyer, U., Wetzel, S., Ioannidis, S.: Distributed Privacy-Preserving Policy Reconciliation. Proceedings of the IEEE International Conference on Communications (ICC), 2007, pp. 1342-1349.
[2] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. Lecture Notes in Computer Science, 2004, pp 1 - 19.
[3] Paillier, P: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Lecture Notes in Computer Science, 1999, pp 223 - 238.

## Implementation

To verify the theoretically determined time-complexity as well as the correctness of our protocols, they were implemented in C++. The protocols and all required components were implemented from scratch using only basic library functions.

### Building Blocks of the PPPR Protocols


Privacy-Preserving Policy Reconciliation Protocols
Privacy-Preserving Tools
Network
Polynomial Representation
Homomorphic Cryptoscheme
Policy Represenation
Rules
Preferences
GNU Multi-Precision Library (GMP)

### Homomorphic Cryptosystem

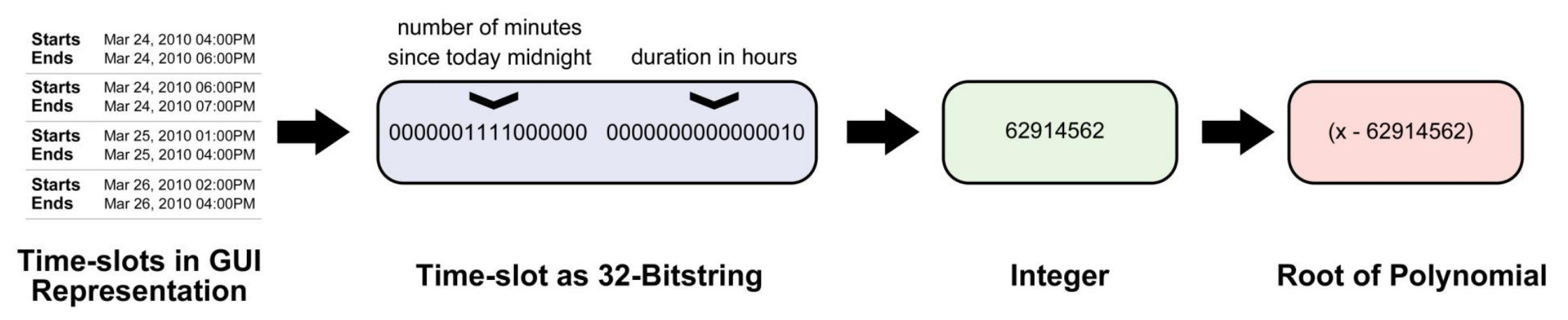
- Allows transformations of the plaintexts by operations on the ciphertexts.
- Our Protocols use the Paillier[3] cryptosystem which, given plaintexts m1, m2 and constant c, has the following properties:
  - $E(m1)^c = E(m1 \cdot c)$
  - $E(m1) \cdot E(m2) = E(m1 + m2)$
- These relations allow the encrypted evaluation of polynomials.
- The cryptosystem was implemented from scratch using the *GNU Multi-Precision* (GMP) library.
  - Key generation and a basic key exchange protocol
  - Encryption and decryption of integers
  - Performance optimizations through pre-computations

### Polynomial Representation

- Polynomials are well-defined by their coefficients.
- Coefficients are stored in an array according to the order of the their term.
- The polynomial class abstracts its internals and offers an interface to modify the polynomial. It includes multiplication with other polynomials, efficient evaluation for plain text and encrypted coefficients, etc.

### Policy Representation

- Policy rules need to be transformable into roots of polynomials.
- Implemented as virtual class. Each potential policy representation provides a function to transform its rules into polynomials, check for equality or set membership.
- Current implementation: Array of bitstrings (ordered by preference).

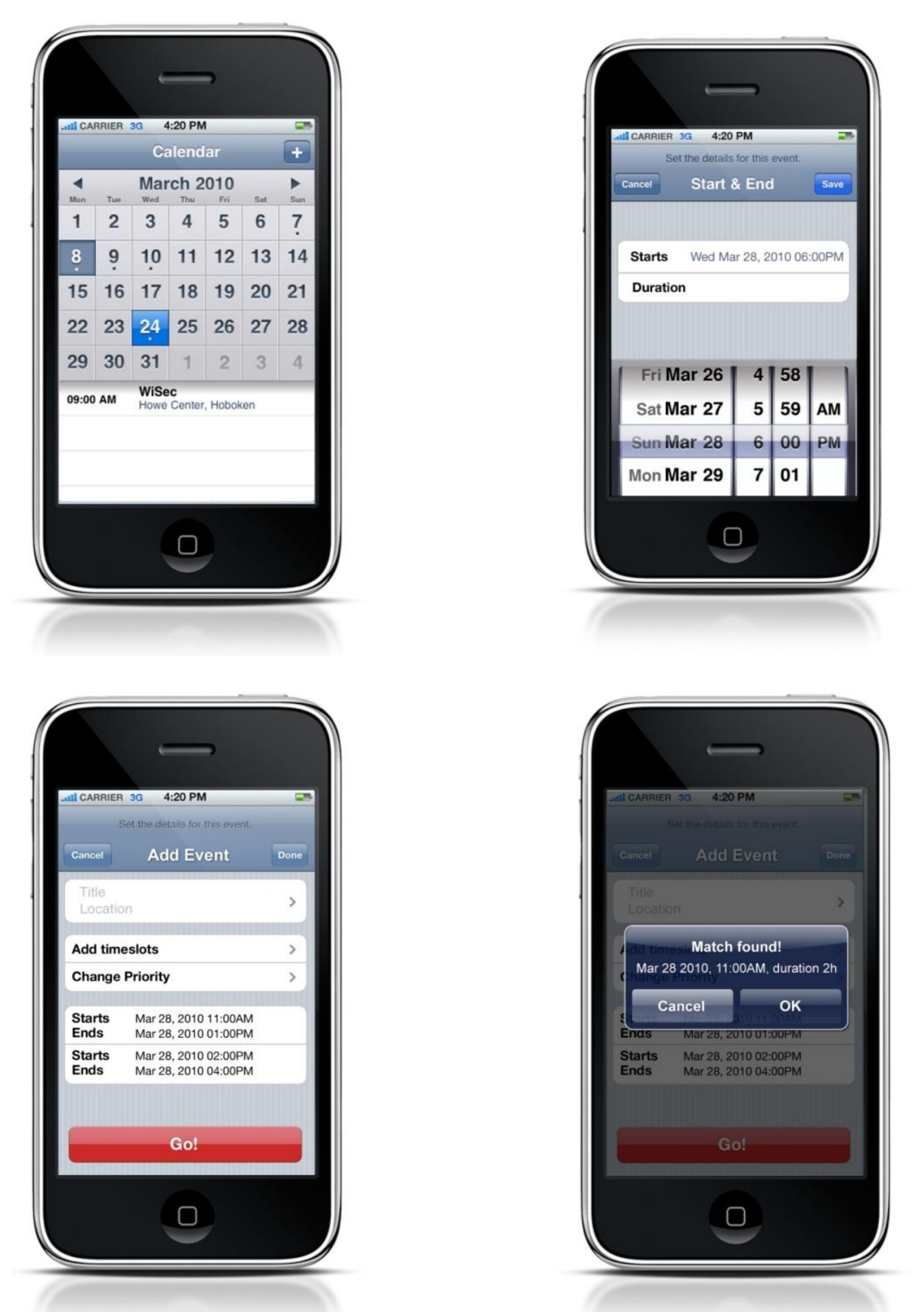

Starts Mar 24, 2010 04:00PM
Ends Mar 24, 2010 06:00PM
Starts Mar 24, 2010 06:00PM
Ends Mar 24, 2010 07:00PM
Starts Mar 25, 2010 01:00PM
Ends Mar 25, 2010 04:00PM
Starts Mar 26, 2010 02:00PM
Ends Mar 26, 2010 04:00PM
number of minutes since today midnight
duration in hours
0000001111000000 0000000000000010
62914562
(x - 62914562)
Time-slots in GUI Representation
Time-slot as 32-Bitstring
Integer
Root of Polynomial

### Networking

- Our PPPR protocols perform a multi-party computation and exchange data with the other parties.
- The design was abstracted in such a way that the communication class can be exchanged as long as it provides a reliable send and receive functionality for byte arrays.
- In appoint, the networking component uses Bluetooth networking provided through Apples GameKit API.

## appoint – the iPhone App

- appoint allows two parties to negotiate the time for a meeting in a privacy-preserving manner.
- Currently, the communication is facilitated using Bluetooth allowing the use of the tool when both parties are in the same physical location.



### Procedure

- Both parties start by specifying a set of free time-slots.
- The time-slots are ordered according to their preference.
- Both parties start the Bluetooth-pairing.
- The negotiation-process is started. The underlying PPPR-protocol assures that the time-slots for each party are kept private.
- The user is notified whether a match was found or not and the possibility of adding it to its calendar is offered.

### Implementation Challenges

- In order to use the PPPR-protocols, a suiteable integer/bitstring representation for the time-slots is required.
- The Linux C++ implementation has to be ported to the iPhone with as little modifications as possible to facilitate future parallel development on both platforms.
- The socket-based communication of the Linux-implementation has to be replaced by Bluetooth networking.
- The limited computing power and resources of the iPhone platform have to be taken into consideration.
- User-Experience: easy way of adding time-slots, results within a few seconds.

### Our Solutions

- Rules are represented as 32-bit integer Z = S||D with integers S (s bits long) and D (d bits long) where 32 = s + d. More sophisticated/finegrained codings are possible.
- The protocol-core runs on the iPhone nearly without modifications when compiled as Objective-C++.
- Bluetooth networking based on the Gamekit API of the iPhone SDK is event-based. A thread safe queue was built to serve as a receive-buffer. The protocol reads the data out of the queue.
- For a practical number of time-slots the performance of the iPhone app is sufficient. Further performance improvements are object of ongoing and future work.