An Implementation of Secure Two-Party Computation for Smartphones with Application to Privacy-Preserving Interest-Cast

Gianpiero Costantino, Fabio Martinelli, Paolo Santi, Dario Amoruso
IIT-CNR, Pisa, Italy
Email: name.surname@iit.cnr.it

Abstract—**In this paper, we present an implementation of the FairPlay framework for secure two-party function computation on Android smartphones, which we call MobileFairPlay. MobileFairPlay allows high-level programming of several secure two-party protocols, including protocols for the Millionaire problem, set intersection, computation of Jaccard similarity coefficient, etc. All these functions are useful in the context of mobile social networks and opportunistic networks, where parties are often requested to exchange sensitive information (list of contacts, interest profiles, etc.) to optimize network operation.**

To demonstrate the feasibility of MobileFairPlay, we present an application to privacy-preserving interest-casting in opportunistic networks, implementing a recently proposed protocol. We tested running times of the implemented protocol on several Android phones, obtaining very reasonable (up to $5sec$) running times. These results clearly promote MobileFairPlay as a feasible security framework for mobile environments.

I. Introduction

Nowadays, crowded places represent for people opportunities to share information deemed interesting. Besides sharing information through traditional, web-based platforms and applications such as Facebook, Twitter, etc., availability of short range radio interfaces in smartphones, tablet PCs, etc. allows for information sharing through direct, opportunistic communication between individuals (typically, using the Bluetooth or WiFi interface). By exploiting mobility of individuals, a piece of relevant information (spanning from a simple advertisement to private documents) can be spread in a geographical domain (e.g., a shopping center, a neighborhood, or even a city) even without connecting to the Internet.

This model of store-carry and forward data to others is known as Opportunistic Networking *(OppNets)* in the literature. Several techniques have been proposed to optimize the information forwarding process in OppNets, including recently proposed approaches based on exploiting information on social ties between individuals to optimize information diffusion [1], [2], [3]. A common features of these approaches is that, before taking a decision on whether sharing an information with an encountered individual, users have to exchange some type of sensitive information, such as history of past encounters [1], [2], interest profiles [3], etc. Considering that the encountered individual is in general a stranger, this exchange of sensitive information (which occurs in plain text in the mentioned approaches) is likely to be deemed unacceptable by the user in real-world scenarios, where privacy concerns play a major role. Thus, the problem of how to optimize the information diffusion process in OppNets without (or only minimally) disclosing sensitive information to the other party has become very important recently.

To our best knowledge, only a few very recent papers deal with the problem of privacy preservation in OppNets, including our recent work on privacy-preserving interest-based message forwarding [4]. Another interesting example is [5], where the authors present a privacy-preserving friendship recommendation system for mobile networks.

While some privacy-preserving approaches to information diffusion and friendship recommendation have been proposed very recently, to our best knowledge the problem of demonstrating *feasibility* of a privacy-preserving approach in mobile environments remains open to date. By “feasibility” here we mean that the privacy-preserving approach can actually be implemented in a mobile platform, and runs in an acceptable time – a few seconds.

In this paper, we present for the first time a *feasible* implementation of a cryptographic framework for Secure Multi-Party computation (the FairPlay framework proposed in [6]) on the Android mobile platform. Secure Multi-Party computation is a very general framework that can be used for privacy-preserving computation of a variety of functions where different parts of the input belong to different users, such as set intersection, comparison of two values, Jaccard similarity coefficient (used, e.g., in [5]), etc. To show feasibility of a specific OppNet application, we consider the privacy-preserving interest-cast approach that we recently proposed [4], and we show that the protocol introduced therein can be executed in a mobile environment with very reasonable running times (a few seconds). In particular, the developed application: 1) find people in Alice’s neighborhood through a Bluetooth scan operation, 2) connect to Bob and discover whether Bob and Alice have similar interest profiles without disclosing sensitive information, and 3) share messages between Alice’s and Bob’s devices only if their profiles are similar. To our best knowledge, the one presented in this paper is the first *feasible* implementation of a privacy-preserving protocol for OppNets in a mobile environment introduced in the literature.

II. Related Work

The problem of implementing privacy-preserving applications in mobile environments has been addressed only in a few recent works. In [7], the authors present an implementation of Secure Multi-party functions for Android smartphones. The applications running on the smartphones communicate through the WiFi interface. While the work of [7] is similar in spirit to ours, the reported running times are much higher than those obtained in our study (ranging from $68sec$ to $> 500sec$ as compared to $\leq 5sec$ in our study). Thus, differently from ours, the approach presented in [7] cannot be considered *feasible* according to the requirements on running time defined in the Introduction. Furthermore, the approach of [7] leaves to the application developer the hard work of translating the desired function into a binary circuit, while ours is a high-level approach in which the developer can easily encode the function through the FairPlay specification language [6], and the hard work of translating the function into boolean is performed once by the FairPlay framework prior to the first application execution (compilation phase).

In [8], De Cristofaro et al. introduce a *Private Contact Discovery* primitive according to which two users can compute their common contacts in a privacy-preserving manner. The authors test the running time of their primitive on different processors, including one used on smartphones. While the reported running times for the smartphone processor seems acceptable ($\leq 7secs$), it is not clear whether the reported time refers to a complete, two party implementation of the primitive, including the time requested to exchange data through a short range radio interface (WiFi or Bluetooth), as done in our study. Furthermore, the one reported in [8] is the implementation of a single secure two-party function, while the implementation of FairPlay on mobile Android devices reported in this study allows the definition of a wide variety of secure two-party functions.

We also briefly mention recently proposed approaches aimed at designing privacy-preserving protocols in opportunistic networks. In [5], the authors propose a technique to recommend friends in a mobile environment based on similarity of contact lists, without disclosing private information to the other party. The approach is based on a secure two-party computation of the Jaccard similarity coefficient between the two contact lists. In [4], we proposed a privacy-preserving implementation of the interest-cast communication primitive for opportunistic networks introduced in [3]. The approach exploits a version of the well-known Yao's protocol for solving the Millionaire problem [9]. In both cases, the authors do not present an actual implementation of the proposed protocols on mobile devices to assess their feasibility. This is what we do in this paper, in particular presenting an implementation of the protocol of [4] on Android smartphones.

III. FairPlay Project

FairPlay [6] is a framework for secure two-party and multi-party [10] computation that allows users to write and run secure functions. In particular, the user writes high-level procedures that are compiled by FairPlay into optimized boolean circuits. Since in mobile environments, and in OppNets in particular, interactions are mostly pair-wise, in the following we focus on the two-party version of FairPlay, which is sufficient to our purposes.

We recall that in secure two-party computation we have two parties (Alice and Bob), each holding some private data x and y, respectively. The goal of secure two-party function computation is allowing Alice and Bob to jointly compute the outcome of a function $f(x, y)$, without disclosing to the other party the own input. The straightforward way to solve the above problem would be to have a Trusted Third Tarty (TTP) to which Alice and Bob securely send the data, and to have the TTP compute $f(x, y)$ and separately send the outcome to Alice and Bob. The business in secure two-party computation amounts to securely compute $f(x, y)$ without the need of a TTP.

FairPlay is shown in [6] to have strong security properties in the context of two-party computation. The framework is shown to be secure against a malicious party; in particular $i)$ a malicious party cannot learn more information about the other party's input than it can learn from a TTP that computes the function; and $ii)$ a malicious party cannot change the output of the computed function. Notice that, as customary in secure two-party computation, there is an asymmetry on the provided security guarantees: in particular, there is no way to prevent Alice from terminating the protocol prematurely, and not sending the outcome of the computation to Bob. This situation can be detected by Bob, but cannot be recovered from.

FairPlay allows the user to specify a secure two-party computation protocol using a high-level language called SFDL. The SFDL code is then compiled into boolean circuits (for both Alice and Bob), which are optimized with the purpose of reducing their size as much as possible. The outcome of this phase is a Java object. Notice that the compiling phase is performed only once before the first execution of the protocol. Secure two-party computation is then realized by having Alice and Bob to exchange their (garbled) circuits through an oblivious transfer protocol, so that both Alice and Bob can eventually know the outcome of $f(x, y)$ by evaluating the respective garbled circuit. Communication between the two parties is realized in FairPlay through establishing a TCP/IP connection.

IV. MobileFairPlay

Two main issues have to be addressed in porting FairPlay to a mobile environment such as Android. First, the Java object computed as outcome of the FairPlay compilation phase has to be made compatible with the JavaVM used in the mobile phone, which is usually slightly different from the standard one. In particular, Android phones use the DalvikVM. The first step in our porting process has then been translating a Java object as produced by FairPlay into a `.dex` file executable on the DalvikVM. Second, FairPlay uses TCP/IP for communication between parties, which is not suitable for setting up and operating a direct radio communication between two smartphones. As customary in opportunistic networking, we used instead the Bluetooth interface for communication between the two parties. The TCP/IP protocol stack used in FairPlay has then been substituted by the Bluetooth protocol stack in MobileFairPlay. In particular, when Alice wants to communicate with Bob, she hooks to the Bluetooth socket of the smartphone (Bluetooth interface must be activated before executing the secure multi-party protocol); a Bluetooth connection request is then sent to Bob, and the communication between the two parties can start once Bob has accepted Alice's connection request.

V. Interest-casting in OppNets

Interest-casting has been recently proposed as a novel forwarding strategy in OppNets [3]. In this section, we briefly describe the principles underlying interest-casting, and the specific interest-casting application we have implemented using MobileFairPlay.

According to [3], each user in an opportunistic network is characterized by an *interest profile*, which is used to drive the information dissemination process within the network. Notice that similar forwarding protocols based on a notion of a user's "social profile" have been recently introduced in the literature [11], [12]. User interests can be modeled as an m-dimensional vector in a common m-dimensional *interest space*, where the number m of interest is typically much smaller than the number n of nodes in the network. More formally, the *interest profile* of user A is defined as:

$$I_A = (a_1, \ldots, a_m) \ ,$$

where $a_i \in [1, max]$ is an integer representing A's interest in the i-th topic of the interest space. Note that interests are expressed as integers in the range $[1, max]$, with 1 representing no interest and max (an arbitrary integer > 0) representing maximum interest[1].

Let S be a user denoted as the message *source*. According to the definition of interest-casting, the message M generated by S (which can be thought of as a piece of information node S wants to share within the network) should be delivered to all nodes in the set $\mathcal{D}(S, \gamma)$, where

$$\mathcal{D}(S, \gamma) = \{U \in \mathcal{N} | sim(U, S) \geq \gamma\} \ ,$$

where $sim(U, S)$ is a similarity metric used to express similarity between a node U and S's interest profiles, with relatively higher similarity values representing relatively more similar interests, and γ is the *relevance threshold* (set by S). Set $\mathcal{D}(S, \gamma)$ is called the set of *relevant destinations*, and in principle it is not known in advance to node S. Instead, set $\mathcal{D}(S, \gamma)$ is implicitly defined by S's interest profile, and by the relevance threshold γ.

[1]The notion of interest profile can be straightforwardly extended to represent also information about a user's habits, such as living in a certain neighborhood, working in a certain place, and so on. For details, see [3].

In the interest-cast implementation presented in the following, we consider the vector-component-wise (vcw) similarity metric defined in [4], which we recall here. Let $S = (s_1, \ldots, s_m)$ and $U = (u_1, \ldots, u_m)$ be the interest profiles of users S and U, respectively. We have:

$$vcw(U, S, \lambda) = \begin{cases} 1 & \text{if } \forall i \in \{1, \ldots, m\}, \ |u_i - s_i| \leq \lambda \\ 0 & \text{otherwise} \end{cases} \ ,$$

where $\lambda \in [0, max]$ is an integer parameter used to narrow/widen the scope of the interest-cast. More specifically, by setting $\gamma = 1$, we have that $\mathcal{D}(S, 1)$ corresponds to the set of all nodes in the network if $\lambda = max$, while $\mathcal{D}(S, 1) = \{S\}$ if $\lambda = 0$.

VI. Privacy-preserving Interest-casting

The forwarding condition in the privacy-preserving version of the interest-casting protocol proposed in [4] requires that message M is copied from node U's buffer to node V's buffer only if a similarity conditions between U and V profiles is satisfied. More specifically, message exchange occurs only if $vcw(U, V, \lambda) = 1$, for some $\lambda \geq 0$. In order to preserve privacy, in [4] it is proposed that, when node U (Alice) and V (Bob) meet, they compute an *estimation* vcw_e of the actual vcw metric, computed using only a *subset* of the interest dimensions (topic) in the profile. Notice that the following property hold: $vcw(U, V, \lambda) = 1 \Rightarrow vcw_e(U, V, \lambda) = 1$, for any number of topics used to compute vcw_e. Unfortunately, the reverse is not true, and it is indeed possible to have $vcw_e(U, V, \lambda) = 1$ even if $vcw(U, V, \lambda) = 1$ (False Positive). FP events should be avoided, since when a FP occurs the piece of information generated by S is delivered to an unintended user, which might lead to disclosure of potentially sensitive information, or simply generate undesired occupation of the buffer at V. The occurrence of FP events can be reduced by increasing the number k of topics used to compute vcw_e (clearly, $Prob(FP) = 0$ when $k = m$). However, the more topics are used to compute vcw_e, the more information is leaked to the other party during the handshaking phase preceding a possible message transfer (see [4] for details). Furthermore, our study reveals that the larger k, the longer the duration of the secure hand-shaking phase. In practice, then, the value of k should be tuned by the application designer to optimize the tradeoff between private information leakage, false positive probability, and protocol running time.

In [4], we proposed to use secure two-party computation to compute $vcw_e(U, V\lambda)$ without disclosing the own interest profile to the other party, except for the information leakage implied by the knowledge of the value of $vcw_e(U, V\lambda)$ at the end of the protocol. More specifically, the protocol proposed in [4] is based on a solution of the well-known "Millionaire Problem" introduced by Yao [9]. We recall that in the Millionaire Problem, two users wants to know whom of them is the richer, without revealing to the other party his/her own amount of money. In particular, the problem is one of computing whether condition $x < y$ holds, where Alice knows only *"x"*, and Bob knows only *"y"*. At the end of the protocol execution, Alice knows only the outcome of the evaluation of condition $x < y$, without knowing y (similarly for Bob).

The protocol proposed in [4] for computing vcw_e is reported in Figure 1, which refers to the case of a single topic used to estimate vcw_e (i.e., $k = 1$). Alice and Bob jointly estimate condition $|i - j| \leq \lambda$, where i represents Alice's interest level in the selected topic and j Bob's interest level in the selected topic. Once Alice and Bob has established similarity of their interests, the data transfer can start. Notice that the similarity condition can be tested by repeating the Millionaire's protocol twice with slightly different input values (see [4] for details).

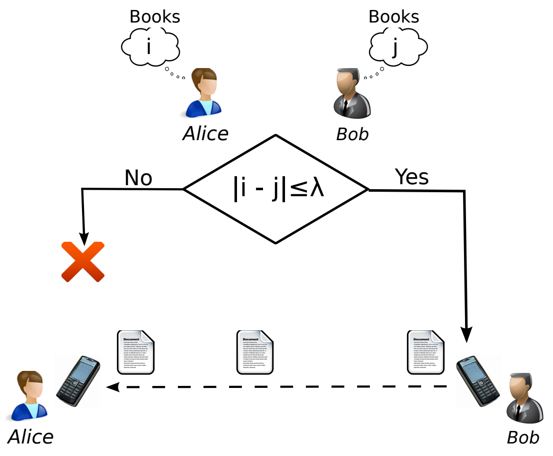


Fig. 1. Protocol flow to discover similarity in a topic (Books).

Notice that, in order to reduce impact of possible Sybil attacks, in [4] it is proposed that the subset of topics used to compute vcw_e is randomly selected by Bob, and not by the protocol initiator Alice.

VII. Interest-casting with MobileFairPlay

In this section, we present the implementation of the basic building block of a privacy-preserving interest-casting application (secure hand-shaking between Alice and Bob to estimate vcw_e, and possible message exchange) realized through MobileFairPlay.

In the developed application, and a user can:

1) set up his own profile regarding different topics;
2) start a new connection with another user and checking if they have similar interests;
3) wait for incoming connections.

Fig. 2 displays the main window of our application.

A. User profile

When the application is run for the first time, the preference window is shown to the user, see Fig. 3. Then, he/she must insert a value for each topic in the window. The possible values that can be inserted are between 1 and 100, where the lowest value means no interest, and the highest value, maximum interest for a topic. Examples of topics in the profile are: *cars, books, movies, sports, television, games,* and others. Finally, the user sets the value of λ required for computing

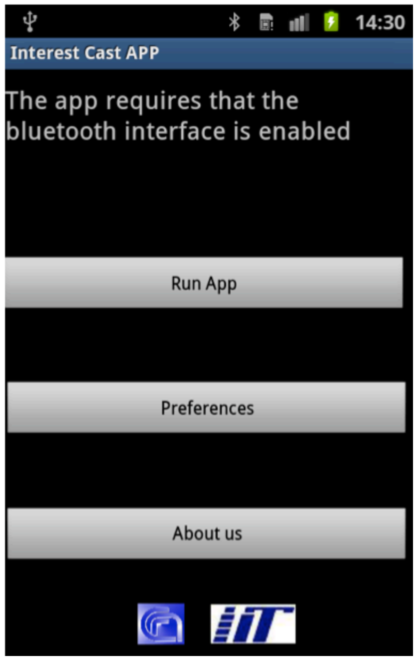
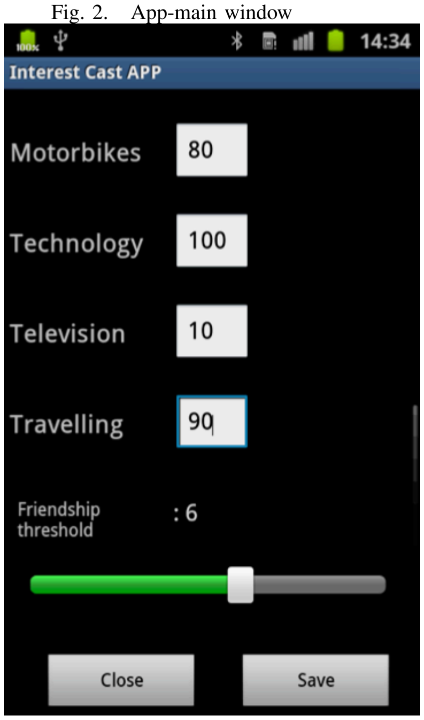

14:30
Interest Cast APP
The app requires that the bluetooth interface is enabled

Fig. 2. App-main window


14:34
Interest Cast APP

Motorbikes	80
Technology	100
Television	10
Travelling	90

Friendship threshold : 6

Fig. 3. User's profile window

similarity metric vcw_e. The value of λ can be set by means of a slide-bar, and accepts values between 0 and 10. When a new incoming request from Alice comes, the metric vcw_e is computed using Bob's λ value (again, this is to reduce the impact of Sybil attacks as noticed in [4]).

B. Privacy-preserving hand-shaking

This part represents the main section of our App. We consider the case in which Alice is carrying on her smartphone, and starts to discovers people around her. Alice decides to connect to another device, which is owned by Bob. Our app manages both the discovery and connection phase with the Bluetooth interface, hence Alice can covers a range of ten-twenty meters around her. Thus, while Bob is waiting for an incoming connection, Alice tries to connect with Bob. After that both devices have been paired, the App works as described in Fig. 4.

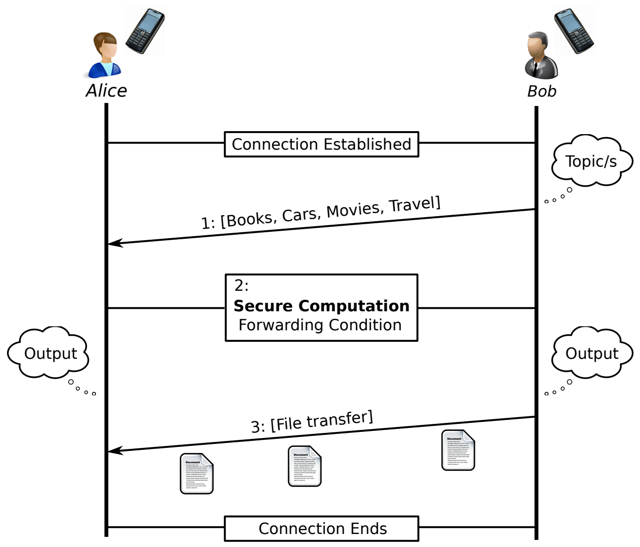


Fig. 4. Protocol steps

1) Topic/s selection: When Alice and Bob have been established a new connection, Bob, who received the connection, randomly selects different topics to verify the forwarding condition. As commented in the previous section, Bob can select a variable number k of topics. In our application, we considered to case of 1 and 4 topics (out of 15 possible topics) used to compute vcw_e. The purpose of using different values of k was mainly estimating the impact of k on the duration of the privacy-preserving hand-shaking phase.

As soon as Alice receives the packet containing the topics selected by Bob, they start verifying the forwarding condition in a secure manner.

2) Secure Computation of the forwarding condition: This is part of the application is realized through MobileFairPlay. The protocol proposed in [4] to compute vcw_e is coded in SFDL. The source code is reported below.

```
program InterestCast {
// Number of topics
const int = 4;
// 4-bit integer as type
type Key = Int<4>;
// 8-bit integer as type
type int = Int<8>;
// Alice's array
type AliceInput = int[4];
// Bob's array, one position is for the
   threshold
type BobInput = int[5];
// Alice's output-array. One position for
    each topic comparison
type AliceOutput = Boolean[4];
// Bob's output-array. One position for
   each topic comparison
type BobOutput = Boolean[4];

type Output = struct {AliceOutput alice,
   BobOutput bob};
type Input = struct {AliceInput alice,
   BobInput bob};

//The function that implements the secure
    forwarding condition
Function Output output(Input input)
{
var int tmp;
var int threshold;
var Key i;

// The threshold is initialised
threshold = input.bob[4];

// For all topics
for (i = 0 to int-1)
{
tmp = (input.bob[i] - input.alice[i]);

//If tmp is less than 0, we negate tmp in
    order to make it positive
if (tmp < 0)
{
        tmp = ~tmp;
}

// if the difference is less than the
   threshold,
// the forwarding condition for the i-th
   topic is verified
if (tmp <= threshold)
{
     output.alice[i] = 1;
     output.bob[i] = 1;
}
else
{
     output.alice[i] = 0;
     output.bob[i] = 0;
}

   }}}
```

Listing 1. Secure Forwarding Condition written in SFDL

So, when the Bluetooth connection is established, Bob and Alice are ready to run the secure function. Bob is the first that runs the function and waits for Alice. She uses the Bluetooth socket that has just established to connect with Bob. At this point, they start to run the secure function according to the secure steps implemented in MobileFairPlay. During this execution, both Bob and Alice use their own value of the selected topic, extracted from the interest profile, to compute vcw_e. However, these values are not sent to the other participant in plain, but they are encoded in the garbled Boolean circuits exchanged through MobileFairPlay. This way, at the end of the hand-shaking phase Alice and Bob only knows the result of jointly computing vcw_e, without knowing the specific interest values of the other party.

We recall that, during the execution of our secure function, the value of λ used is the one set by Bob in his own preference window. So, at the end of the hand-shaking phase, the result is positive (and file transfer can start) if and only if, for all topics selected by Bob, the similarity condition is verified. Only in this case, Bob accepts Alice as a person with similar interests, and exchange files with her.

3) Files Transfer: Once the hand-shaking phase has established that Alice and Bob have similar interests, Bob sends his files to her. In fact, Bob stores in his smartphone a folder containing the files which he received in the past from other users with similar interests, or that he decided to share because deemed of interest for other people.

Our developed application allows exchanging files of any kind and any extension; in fact, raw bytes are exchanged during the file exchange phase, allowing to transfer files of arbitrary format. In our tests, we have successfully transfered text files (`.txt`), pdf files, image files (`.jpg`), etc. A file can represent a movies-advertisement regarding all cinemas in a city, an advertisement of a rock music-concert, or any other kind of information. We recall that the main idea of interest-casting is that, since Bob and Alice share a similar interest, the files that Bob has are likely to be of interest for Alice too.

VIII. Computational-time evaluation

Here, we present the results —in terms of execution-time— of two main studies, regarding:

- compilation of the SFDL code, and
- running of the secure function (hand-shaking phase).

The results were obtaining testing our application on five different model of smartphones currently present in the market. Table I reports the specification of the models considered in our evaluation.

A. Compiling the secure function

The current version of the APP required that when it is run for the first time, the application creates the boolean circuits that are executed during the secure hand-shaking phase. The time required to convert the function into boolean circuits depends on the function that is written as SFDL. In this analysis, we compare two functions: one that evaluates vcw_e based on a single topic, and the other one in which the compared topics are four. The code given as input to the MobileFairPlay compiler is that one in code listed in 1, which refers to the case of four topics.

Smartphone	**Time (ms)**
Samsung Galaxy S2	382
Samsung S-plus	446
Samsung S	492
Lg Optimus Dual	449
Htc Desire	489

TABLE II
Compilation time required for one topic function

Smartphone	**Time (ms)**
Samsung Galaxy S2	4471
Samsung S-plus	5347
Samsung S	6605
Lg Optimus Dual	5352
Htc Desire	6512

TABLE III
Compilation time required for four topics function

In Table II and III, the time that each device requires to compile the secure function is shown. In the first case, the circuits are quickly created. Instead, the time to convert the function of four topics is higher. In this phase, the most important requirement is the smartphone hardware. In fact, those ones with higher frequency-clock gets a lower time for the compilation. Notice that the compilation time displays a super-linear increase with the number k of topics used to evaluate vcw_e, and it is quite high in the case of four topics. However, these results do not impair the feasibility of the developed interest-cast application, since compilation is performed only once, during the first execution of the application (it can be considered as part of the application installation time).

B. Running the secure function (hand-shaking)

In this section we evaluate the time that Alice and Bob need to run the secure function, i.e., the duration of the hand-shaking phase.

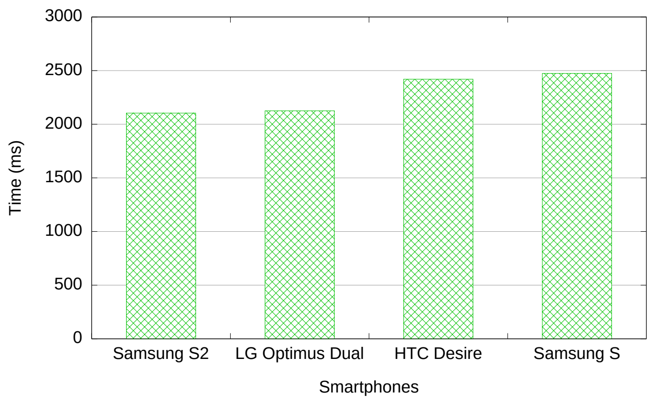


Fig. 5. One topic: Bob running time of the secure function

Smartphone	**CPU**	**RAM**	**Bluetooth Ver.**	**Android O.S.**
Samsung Galaxy S2	Dual-core 1228 MHz	1 GB	3.0	2.3.6
Samsung S-plus	Single-core 1443 MHz	512 MB	3.0	2.3.5
Samsung S	Single-core 1024 MHz	512 MB	3.0	2.3.3
Lg Optimus Dual	Dual-core 1024 MHz	512 MB	2.1	2.3.4
Htc Desire	Single-core 1024 MHz	576 MB	2.1	2.2.3

TABLE I
SMARTPHONES USED FOR TESTING THE INTEREST-CAST APPLICATION

Fig. 5 shows the time needed to run the secure function in the case of one topic comparison. This time includes the time needed to exchange the garbled boolean circuits through the Bluetooth interface, and to compute the output of vcw_e. The results are obtained considering the case in which the role of Alice is kept constant. In fact, while Bob is run four times with four different smartphones, Alice is instead always run on the Samsung Galaxy Plus.

Fig. 6 considers the opposite case. Now, the role of Bob is kept on the Samsung Galaxy Plus, while Alice is run on the other devices. As expected, we observe that in both cases the higher the computational power of the smartphone, the lower the time required to execute the secure function evaluation. Furthermore, we notice that by varying the role of Alice, the computational power of the smartphone has a stronger effect on the running time. This is because in the case of two party computation, Alice runs the heavier role, and she benefits more of a powerful smartphone.

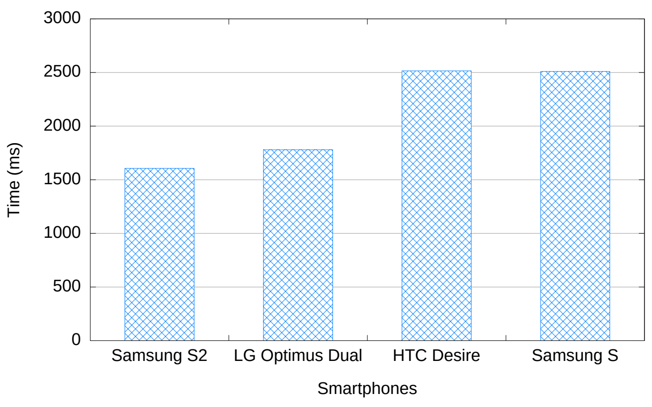


Fig. 6. One topic: Alice Running Time of the secure function

We then proceeded to test the secure function in the case of four topics comparison. In Fig. 7 Alice is run on the Samsung Galaxy Plus, while Bob uses a different smartphone each time. Even in this case, the behavior is similar to the previous case. As expected, the best result is obtained with the most powerful device. Figure 8 reports running times when Bob is run on the Samsung Galaxy Plus, while Alice uses different smartphones. It is interesting to observe that, while a clearer benefit of Alice using a more powerful device is still perceivable, the relative advantage provided by a powerful smartphone is smaller than in the case of single topic computation. This is due to the fact that with four topics the exchanged garbled Boolean circuits are larger, and the time required to exchange the circuits over the Bluetooth link (which is largely independent of the smartphone CPUs) tends to dominate the overall running time. Finally, we observe that, differently from the compilation time, the duration of the secure function computation (hand-shaking) is *sub-linear* with the number k of topics used to compute vcw_e. For instance, considering the case of Alice running on Samsung Galaxy Plus and Bob running on Samsung Galaxy S2, we have a running time of $2.1sec$ with one topic, and of $3.7sec$ with four topics. These results are then very encouraging for the scalability of the designed interest-casting application to larger interest profiles.

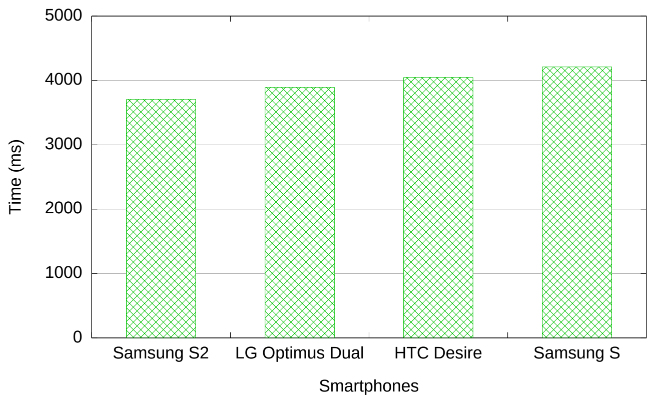


Fig. 7. Four topics: Bob Running Time of the secure function

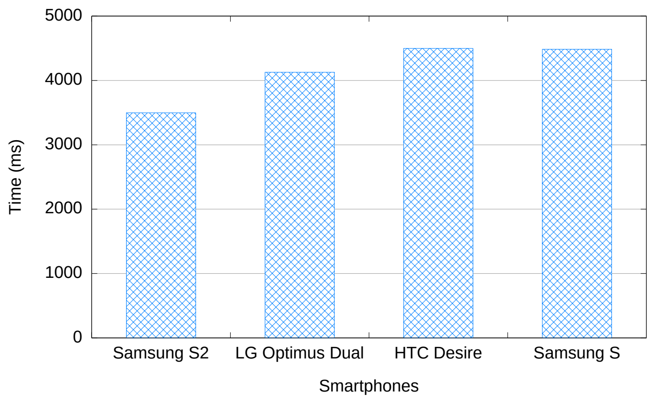


Fig. 8. Four topics: Alice Running Time of the secure function

Summarizing, our results clearly show that protocols belonging to the class of Secure Two-Party Computation nowadays can be efficiently executed on mobile devices. Although the running time required is not negligible, it always turned to be below $5 sec$ for any pair of smartphones considered in our study. We can expect running times to become even lower as computational and communication power of smartphones increase.

IX. Conclusion

In this work, we have presented for the first time a framework, derived from project FairPlay [6] and which we call MobileFairPlay, for secure two-party function computation in mobile environments based on the Android operating system. MobileFairPlay inherits the nice features of FairPlay, such as ease of use (high-level protocol programming) and efficiency, and port them in a mobile environment.

To assess the effectiveness of MobileFairPlay, we have implemented and extensively tested a privacy preserving interest-cast application for opportunistic networks. Our tests, performed using five different smartphones, have revealed that MobileFairPlay is a *feasible* framework for mobile environments: the running time of our application resulted always below $5 sec$, independently of the smartphones used by the two parties involved in secure function computation.

For future work, we plan to implement other secure two-party functions using MobileFairPlay and to test them, in order to further assess MobileFairPlay as a framework for efficient secure two-party execution in mobile environments. We also plan to extend MobileFairPlay to support other mobile platforms, such as iOS and Symbian.

References

[1] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *ACM MobiHoc*, 2007.

[2] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay tolerant networks," in *ACM MobiHoc*, 2008.

[3] A. Mei, G. Morabito, P. Santi, and J. Stefa, "Social-aware stateless forwarding in pocket switched networks," in *IEEE Infocom*, 2011.

[4] G. Costantino, F. Martinelli, and P. Santi, "Privacy-preserving interest-casting in opportunistic networks," in *IEEE Wireless Communication and Networking Conference (WCNC)*, 2012.

[5] E. Baglioni, L. Becchetti, L. Bergamini, U. Colesanti, L. Filipponi, A. Vitaletti, and G. Persiano, "A lightweight privacy-preserving sms-based recommendation system for mobile users," in *ACM RecSys*, 2010.

[6] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay&#8212;a secure two-party computation system," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 20–20. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251375.1251395

[7] Y. Huang, P. Chapman, and D. Evans, "Privacy-preserving applications on smartphones," in *Proceedings of the 6th USENIX conference on Hot topics in security*, ser. HotSec'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028040.2028044

[8] E. De Cristofaro, M. Manulis, and B. Poettering, "Private discovery of common social contacts," in *Proceedings of the 9th international conference on Applied cryptography and network security*, ser. ACNS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 147–165. [Online]. Available: http://dl.acm.org/citation.cfm?id=2025968.2025980

[9] C. Andrew and C. Yao, "Protocols for secure computations," in *23rd IEEE Symposium on FOCS*, 1982, pp. 160 –164.

[10] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 257–266. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455804

[11] J. Wu and Y. Wang, "Social-feature based multi-path routing in delay tolerant networks," in *IEEE Infocom*, 2012.

[12] D. Eppstein, M. Goodrich, M. Loffler, D. Strash, and L. Trott, "Category-based routing in social networks: Membership dimension and the small-world phenomenon," in *IEEE Conf. on Computational Aspects in Social Networks (CASoN)*, 2011.