# Private Location Verification

Vadym Fedyukovych

September 15, 2018

**Abstract**

We present location verification protocol and implementation that allows for location privacy. With a Schnorr-like protocol, we verify that location committed is close enough to another known location. Protocol was implemented with Crypto++ library.

# 1 Introduction

Need for location privacy. Known results and state of the art. SNARK-based and 'older' interactive proofs-based solutions.

## 1.1 Our contribution

# 2 Protocol

## 2.1 Definitions

## 2.2 Notations

Airdrop location $(x_l, y_l)$ available in clear, node location $(x_n, y_n)$ hidden (committed), acceptable maximum distance $d$ from node to airdrop, setup: group description and parameters, group elements for making commitments, initial message, challenge and responces of a Schnorr-like protocol.

## 2.3 Interactive proof

Interactive argument system about integers is designed with a group of a hidden order [], that is, order of the group is not available to the Prover. Inequality (not far from) statement is converted into equality with 4-squares Lagrange theorem (Lipmaa). Schnorr proof was extended into a proof systems for polynomial relations with polynomials of higher degree in challenge for a number of applications [3, 2]. A comparable proof system for integers was introduced at [4].

### 2.4 Proof setup

Proof system for relations about integers is well described at Idemix documentation [1]

Multiplicative group of residue classes, RSA-like modulus.

Common input of Prover and Verifier is commitment $s_U$ to node location (1), airdrop location $(x_l, y_l)$, threshold $d^2$, and parameres $(g, g_x, g_y, g_r, h_j)$:

$$s_U = g_x^{x_n} g_y^{y_n} g^r \tag{1}$$

Private input of Prover is node location $(x_n, y_n)$ and location commitment randomness $r$, four numbers $\{a_j\}$ calculated with Rabin-Shallit algorithm according to (2). Statement being proved is

$$d^2 - ((x_n - x_l)^2 + (y_n - y_l)^2) = \sum_{j=1}^{4} a_j^2 \tag{2}$$

Protocol runs as follows:

1. Prover picks random $\alpha_j, \eta, \gamma, \beta_x, \beta_y, \beta_r, \rho_0, \rho_1$, computes $f_0, f_1$, and sends initial commitments $b_0, b_1, t_a, t_n$:

$$f_0 = -(\beta_x^2 + \beta_y^2) - \sum_{j=1}^{4} \alpha_j^2,\ f_1 = -2((x_n - x_l)\beta_x + (y_n - y_l)\beta_y) - 2\sum_{j=1}^{4} a_j \alpha_j \tag{3}$$

$$t_n = g_x^{\beta_x} g_y^{\beta_y} g^{\beta_r},\ s_a = g^{\gamma} \prod_{j=1}^{4} h_j^{a_j},\ t_a = g^{\eta} \prod_{j=1}^{4} h_j^{\alpha_j},\ b_0 = g^{f_0} g_r^{\rho_0},\ b_1 = g^{f_1} g_r^{\rho_1} \tag{4}$$

2. Verifier chooses and sends his challenge $c$
3. Prover computes and sends responses

$$X_n = cx_n + \beta_x,\ Y_n = cy_n + \beta_y,\ R = cr + \beta_r \tag{5}$$

$$A_j = ca_j + \alpha_j,\ R_a = c\gamma + \eta,\ R_d = c\rho_1 + \rho_0 \tag{6}$$

4. Verifier accepts if

$$g_x^{X_n} g_y^{Y_n} g^R s_U^{-c} = t_n, \quad g^{R_a} (\prod_{j=1}^{4} h_j^{A_j}) s_a^{-c} = t_a \tag{7}$$

$$g^{c^2 d^2 - ((X_n - cx_l)^2 + (Y_n - cy_l)^2) - (A_1^2 + A_2^2 + A_3^2 + A_4^2)} g_r^{R_d} = b_1^c b_0 \tag{8}$$

Figure 1: Private location verification protocol

### 2.5 Security properties

## 3 Implementation

We have this protocol implemented on top of Crypto++ library [1] serving as a bignumbers backend.

Producing four-squares witness [5] is a work in progress, and is not a part of the protocol reported. To facilitate larger proof-of-concept application, a temporary approximate solution was introduced producing four squares.

## 4 Discussion and Conclusion

## References


[1] Jan Camenisch. Specification of the identity mixer cryptographic library RZ 3730 version 2.3.0, 2010.

[2] Giovanni Di Crescenzo and Vadym Fedyukovych. Zero-knowledge proofs via polynomial representations. In *Proceedings of the 37th International Conference on Mathematical Foundations of Computer Science*, pages 335–347, 2012.

[3] Vadym Fedyukovych. An argument for Hamiltonicity. Cryptology ePrint Archive, Report 2008/363, 2008.

[4] Vadym Fedyukovych. Proving outcome of private statistical signal testing. In *Statistical Methods of Signal and Data Processing*, pages 172–175, 2010.

[5] Paul Pollack and Enrique Trevino. Finding four squares in Lagrange's theorem.


[1] `https://cryptopp.com/`