PQC: R-Propping of a Simple Oblivious Transfer

Pedro Hecht

Information Security Master, School of Economic Sciences,
School of Exact and Natural Sciences and Engineering School (ENAP-FCE),
University of Buenos Aires, Av. Cordoba 2122 2nd Floor,
CABA C1120AAP, República Argentina
phecht@dc.uba.ar, qubit101@gmail.com

Abstract. Post-quantum cryptography (PQC) is nowadays a very active research field [1]. We follow a non-standard way to achieve it, taking any common protocol and replacing arithmetic with GF(2^8) field operations, a procedure defined as R-Propping [2-7]. The resulting protocol security relies on the intractability of a generalized discrete log problem, combined with the power sets of algebraic ring extension tensors and resilience to quantum and algebraic attacks. Oblivious Transfer (OT) is a keystone for Secure Multiparty Computing (SMPC) [8], one of the most pursued cryptographic areas. It is a critical issue to develop a fast OT solution because of its intensive use in many protocols. Here, we adopt the simple OT protocol developed by Chou and Orlandi [9] as the base model to be propped. Our solution is fully scalable to achieve quantum and classical security levels as needed. We present a step-by-step numerical example of the proposed protocol.

Keywords: Post-quantum cryptography, combinatorial group theory, finite fields, R-propping, secure multiparty computing, oblivious transfer

1. Introduction

1.1. PQC Proposals Based on Combinatorial Group Theory

Besides currently, NIST evaluated PQC solutions like code-based, hash-based, multi quadratic, or lattice-based cryptography, there remain overlooked solutions belonging to non-commutative (NCC) and non-associative (NAC) algebraic cryptography. The general structure of these solutions relies on one-way trapdoor functions (OWTF) extracted from the combinatorial group theory [10]. Here we use an Algebraic Ring Extension (AER) as described in [2]. This algebraic structure is not fully explored, more theoretical research is needed.

1.2. The motivation of the present work

R-propping consists of replacing numerical field operations with algebraic operations using the AES field [11], but any other polynomial field extension could be used as long the generalized discrete logarithm problem remains hard to solve and no brute-force exploration could be performed. As a benefit, no big number is needed and eradicating the critical dependency on pseudo-random generators that affects protocols whose security relies on big prime numbers.

Otherwise, it is of utmost interest to provide an R-Propped OT solution, one of the fundamental building blocks of SMPC. The so-called simplest solution of Chou and Orlandi is a direct extension of the Diffie-Hellman key exchange protocol, and their security is assured as long the computational Diffie-Hellman problem is hard to solve. In their paper, the authors describe in full detail the security concerns of the proposed solution [9].

2. Original Chou-Orlandi OT solution

Here we present the $\binom{2}{1}$ –OT protocol, easily expandable to $\binom{n}{1}$ –OT (n>2) which is discussed in [9]. The generic 1-out-of-2 OT scheme is described in Figure 1:

Figure 1. Oblivious Transfer begins with Alice (the sender) posting two messages m_0 and m_1. Bob (the receiver) privately and randomly selects a bit b and recovers the corresponding message m_b, but nothing more. Alice does not acknowledge the selected bit from Bob and Bob ignores the unselected message.

The actual implementation of the original work of Chou and Orlandi is presented at the next point and schematical in Figure 2.

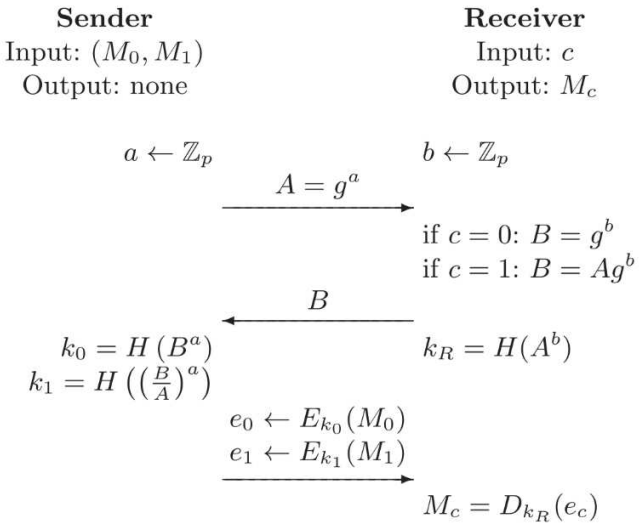

Figure 2. The original Chou-Orlandi simplest proposal. A slightly modified explanation of the protocol is described below. The Figure was extracted from their paper [9].

3. Chou-Orlandi OT implementation.

3.1. **SETUP**. The participating entities agree to use a prime p and the cyclic multiplicative group $G=Z_p$ together with a generator $<g>$. They also agree on a hash function H() and a cipher method $E_{key}( )$. The authors propose the twisted Edwards group, SHA-256 as the hash function and Curve25519 as the appropriate ECC encryption scheme. Numerical operations are reduced mod p.

3.2. **PRIVATE KEYS**. Both sender and receiver select at random integers a and b belonging to Z_p as their private exponents.

3.3. **SENDER MESSAGE PAIR**. The sender prepares privately two messages (M_0, M_1).

3.4. **RECEIVER BIT CHOICE**. The receiver chooses a random private bit c.

3.5. **SENDER TOKEN A.** $A=g^a$ and deliver it to the receiver.

3.6. **RECEIVER TOKEN B.** $B=g^b$ if c=0 and B=Ag^b if c=1 and deliver *B* to the sender.

3.7. **RECEIVER DECIPHER KEY.** He prepares $k_R=H(A^b)$.

3.8. **SENDER CIPHER KEYS.** He prepares two keys $k_0=H(B^a)$ and $k_1=H((B/A)^a)$.

3.9. **THE SENDER ENCRYPTS AND DELIVERS BOTH MESSAGES.** Using the two keys, he ciphers $e_0=E_{k0}(M_0)$ and $e_1=E_{k1}(M_1)$ and sends both.

3.10. **THE RECEIVER RECOVERS ONE MESSAGE**. Using his decipher key, he recovers only one of the message pair computing $M_c=D_{kR}(e_c)$.

4. R-Propped Chou-Orlandi OT adaptation.

This work is mainly based on the preceding description but adapted in a simpler environment. Security was a pursued goal to assure its adaptability to real-world scenarios. The main differences are:

4.1. Instead of numbers belonging to the Z_p group, we work with d-dimensional tensors in an AER with GF(2^8) operations. Specifically, we choose to use as the working group, cyclic multiplicative subgroups of non-singular tensors as elsewhere described [2-7]. With that restriction we detected some generators of high multiplicative order, which could be employed with two purposes: obtain easily inverse tensors and foil systematic exploration of the keyspace. The tabulated generators presented al Table 1 from [6] are transformed into actual session generators rising them to random exponents.

4.2. As a hash function, we choose a double SHA-256 with reversal, an original proposal that is described in Figure 3. For greater security, we suggest replacing SHA-256 with SHA3-512.

4.3. The encryption function is a VERNAM cipher with 256-bit plain text and equally length unstructured random key, to attain perfect secrecy. As actual keys are hash results, the double variant described above tends to foil collision attacks and simulates the required randomness for ciphering procedure.

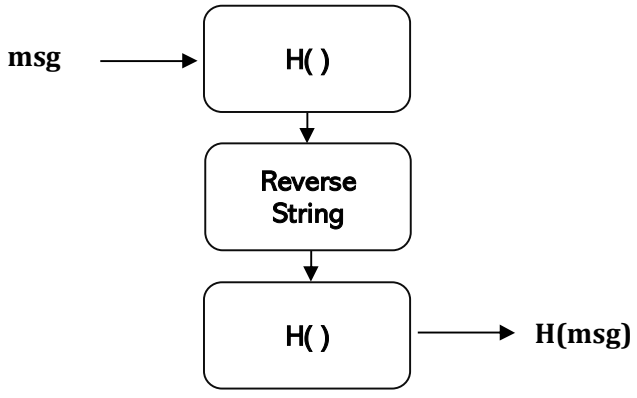

Figure 3. The Actual hash used for R-Propped Chou-Orlandi OT. Here H() represents any secure hash function, we choose SHA-256. The double-pass with reversal was designed to strengthen the randomness needed for the encryption step.

5. Step-By-Step Example

Here we show a dim=3 program for interpreted Mathematica 12 language. Running as-is on an Intel®Core™i5-5200U CPU 2.20 GHz the registered mean session time was 1.19 s. The dim=3 generator was selected from Table 1. published in [6]. The whole sequence follows point 3. Description (Figure 2.).

```
In[ ]:=
Print["............................................"]
Print[" R-PROPPING OF 1-out-of 2 Oblivious Transfer"]
Print["......................SETUP..................."]
dim = 3; Print["tensor dimension = ", dim];
period = 2^24 - 1; Print["tensor period = ", period];
g0 = {{158, 215, 6}, {216, 221, 53}, {45, 119, 206}};
Print["original tensor = ", MatrixForm[g0]];
n = RandomInteger[{2, period - 2}]; Print["random power(n) = ", n];
g = TFastPower[g0, n]; Print["public generator <g> = ", MatrixForm[g]];
Print["hashing = SHA-256 (doubled w/reversal)"];
Print["cipher  = VERNAM"];
Print["................ALICE PRIVATE................"]
x = RandomInteger[{2, period - 2}]; Print["ALICE secret power(x) = ", x];
m0 = "First hidden string"; Print["ALICE message m0 = ", m0];
m1 = "Second hidden string"; Print["ALICE message m1 = ", m1];
Print["................ALICE TOKEN................."]
X = TFastPower[g, x]; Print["ALICE token (X) = ", MatrixForm[X]];
Print["................BOB PRIVATE ................."]
y = RandomInteger[{2, period - 2}]; Print["BOB secret power(y)= ", y];
b = Mod[RandomInteger[{2, period - 2}], 2]; Print["BOB random bit choice (b) = ", b];
t = TFastPower[X, y];
K_b = dhash[ToString[t]]; Print["BOB decryption key (K_b) =", K_b];
Print["................BOB TOKEN..................."]
If[b == 0, Y = TFastPower[g, y], Y = TProd[X, Y]];
Print["BOB token (Y) =", MatrixForm[Y]];
Print["........ALICE CIPHERS BOTH FOR BOB..........."]
d = TFastPower[Y, x];
K_0 = dhash[ToString[d]]; Print["encryption key (K_0) =", K_0];
Xinv = TFastPower[X, period - 1];
s = TProd[Y, Xinv]; f = TFastPower[s, x];
K_1 = dhash[ToString[f]]; Print["encryption key (K_1) =", K_1];
c_0 = BitXor[FromDigits[K_0], FromDigits[dhash[ToString[m0]]]];
Print["cipher c_0 = ", BaseForm[c_0, 16]]
c_1 = BitXor[FromDigits[K_1], FromDigits[dhash[ToString[m1]]]];
Print["cipher c_1 = ", BaseForm[c_1, 16]]
Print["........BOB DECIPHERS ONLY ONE............."]
If[BitXor[FromDigits[K_b], c_0] == FromDigits[dhash[ToString[m0]]],
  Print["m0 recovered with K_b key from c_0"], Print["m0 failure"]];
If[BitXor[FromDigits[K_b], c_1] == FromDigits[dhash[ToString[m1]]],
  Print["m1 recovered with K_b key from c_1"], Print["m1 failure"]];
Print["............................................"]
```

Figure 4. Small example program of the defined protocol. In a real-world application, dim =7 or greater should be used to get reasonable security.

Upon request to the author, full source code is available, including the newly defined functions for this protocol. The actual output of this code is presented below.

```
.........................................
 R-PROPPING OF 1-out-of 2 Oblivious Transfer
.....................SETUP.................
tensor dimension = 3
tensor period = 16 777 215
                  ⎛158  215   6 ⎞
original tensor = ⎜216  221  53 ⎟
                  ⎝ 45  119 206 ⎠
random power(n) = 8 717 820
                          ⎛ 42   51   37 ⎞
public generator <g> =    ⎜229  142  151 ⎟
                          ⎝195   69  191 ⎠
hashing = SHA-256 (doubled w/reversal)
cipher  = VERNAM
.................ALICE PRIVATE.............
ALICE secret power(x) = 16 622 742
ALICE message m0 = First hidden string
ALICE message m1 = Second hidden string
.................ALICE TOKEN...............
                  ⎛ 13  254   7 ⎞
ALICE token (X) = ⎜133  172 197 ⎟
                  ⎝125  229 225 ⎠
.................BOB PRIVATE ..............
BOB secret power(y)= 2 691 032
BOB random bit choice (b) = 1
BOB decryption key (K_b) =
 352ea59b1f93520070d608ea7ccaa0aaa607dcd2b50d8e42fba65010acf740a6
.................BOB TOKEN.................
                ⎛216  91   31 ⎞
BOB token (Y) = ⎜ 30  205   5 ⎟
                ⎝ 93  151 239 ⎠
.........ALICE CIPHERS BOTH FOR BOB..........
encryption key (K_0) =
 11f7db5f92e13ac209df6f6fd54bf33ce5123981a6c6edb19d7e693a412c58fa
encryption key (K_1) =
 070d41db1991e3e13122dc5ea350f1fbe8a5eb322e0d771ce9d599ca79aa0edc
cipher c_0 = ea311e90ba2d543ec9a5c0e75cb3b5911152b755043fade17fea2_16
cipher c_1 = cd960cde16b09953c04004fdf83e401b4ffe90d0a5c15a188592c_16
.........BOB DECIPHERS ONLY ONE..............
m0 failure
m1 recovered with K_b key from c_1
.........................................
```

Figure 4. All strings are in Hexadecimal. This is the output produced by a random session of the described source code.

6. The cryptographic security of the R-Propped B-D protocol

The quantum and classical security were first described at [6] and here repeated in Table 1.

Tensor dimension	$<G_0>$ base generator	cyclic period $\lvert<G>\rvert$	Classical Security (bits)	[Grover] Quantum Security (bits)
3	G3	$2^{24} - 1 = 16777215$	24	12
4	G4	$2^{32} - 1 = 4294967295$	32	16
7	G7	$2^{96} - 1 = 7.92 \times 10^{28}$	96	48
10	G10	$2^{112} - 1 = 5.19 \times 10^{33}$	112	56
12	G12	$2^{160} - 1 = 1.46 \times 10^{48}$	160	80

Table 1. Expected security of increasing size of private keys subject to classical and quantum attacks. Depending on the situation, should be chosen original generators like G7 or above from Table 1. of the [6] paper. At any case, any random power of the base generator should be used as the actual generator of the protocol.

The IND-CPA2 semantic security [12] is assured as members of the <g> set are indistinguishable from random tensors of the same size. More arguments and statistical evidence of tensor structures are provided at [3,4].

5 Conclusions

We present a PQC adaptation of a fast Chou-Orlandi OT. Practical parameters are discussed, and they solve the central question with different security levels.

References

1. D. J. Bernstein, T. Lange, "Post-Quantum Cryptography", Nature, 549:188-194, 2017
2. P. Hecht, "Algebraic Extension Ring Framework for Non-Commutative Asymmetric Cryptography",ArXiv Cryptography and Security (cs.CR), https://arxiv.org/abs/2002.08343, 4pp (2020), DOI: 10.13140/RG.2.2.25826.56002
3. P. Hecht, "PQC: R-Propping of Public-Key Cryptosystems Using Polynomials over Non-commutative Algebraic Extension Rings", https://eprint.iacr.org/2020/1102, 10pp (2020) DOI: 10.13140/RG.2.2.25826.56002
4. P. Hecht, "R-Propping of HK17: Upgrade for a Detached Proposal of NIST PQC First Round Survey", https://eprint.iacr.org/2020/1217, 7pp (2020) DOI: 10.13140/RG.2.2.31287.96163
5. P. Hecht, "PQC: R-Propping of Burmester-Desmedt Conference Key Distribution System", https://eprint.iacr.org/2021/024, DOI:10.13140/RG.2.2.22638.43846 (2021)
6. P. Hecht, "PQC: R-Propping of a New Group-Based Digital Signature", https://eprint.iacr.org/2021/270, DOI: 10.13140/RG.2.2.35795.91683 (2021)
7. P. Hecht, "PQC: R-Propping a Chaotic Cellular Automata", https://eprint.iacr.org/2021/672, DOI: 10.13140/RG.2.2.11309.61924 (2021)
8. P. Bogetoft et al, "Secure multiparty computation goes live", https://eprint.iacr.org/2008/068.pdf, (2008)
9. T. Chou, C. Orlandi, "The Simplest Protocol for Oblivious Transfer", https://eprint.iacr.org/2015/267.pdf, (2015)
10. A. Myasnikov, V. Shpilrain, A. Ushakov, Non-commutative Cryptography and Complexity of Group-theoretic Problems, Mathematical Surveys and Monographs, AMS Volume 177, 2011
11. FIPS PUB 197: the official AES standard, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, (2001)
12. E. Kiltz, J. Malone-Lee, A General Construction of IND-CCA2 Secure Public Key Encryption, ruhr-uni-bochum.de/Eike.Kiltz/papers/general_cca2.ps